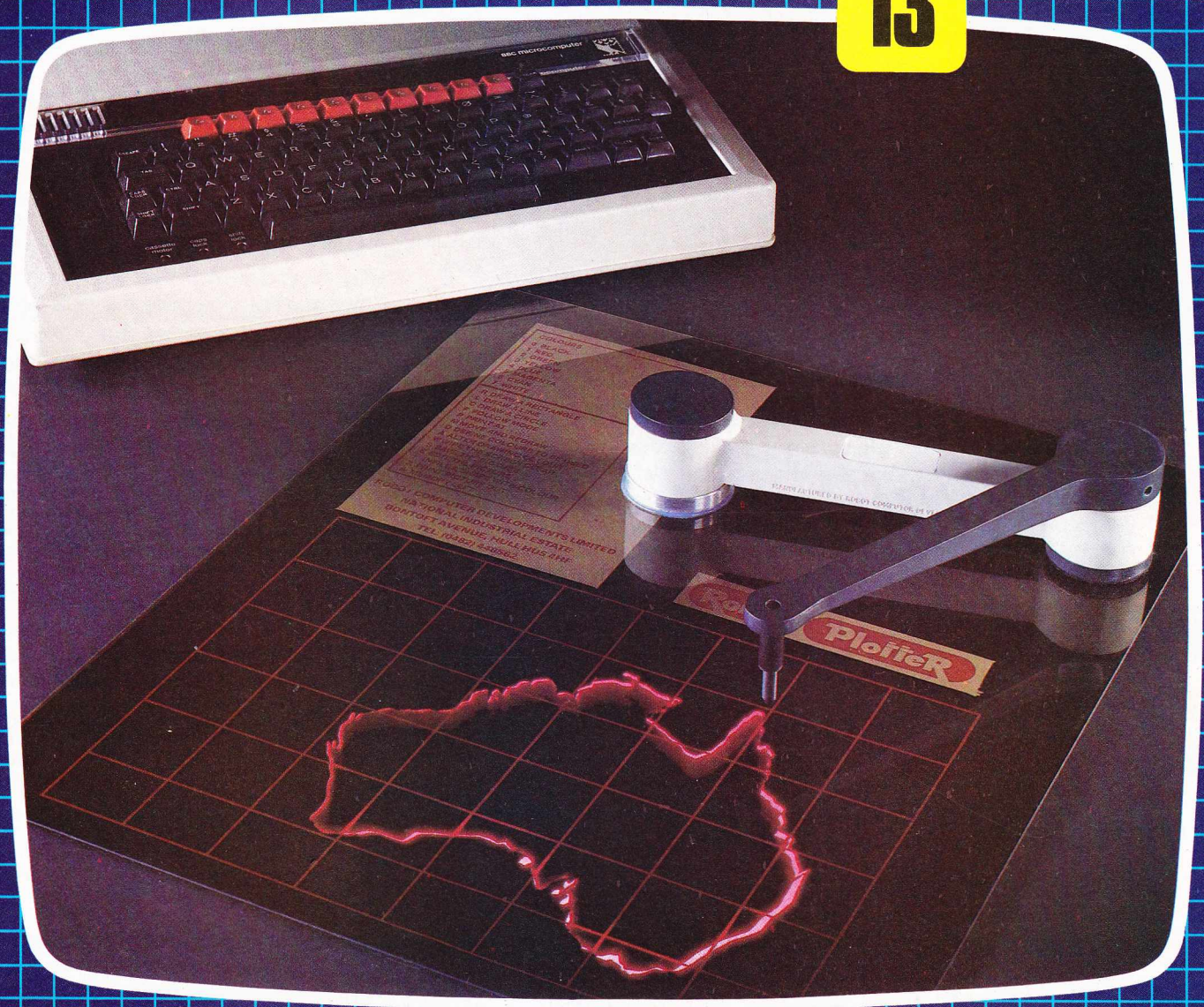
MICROCOMPUTADOR CURSO PRÁTICO

TIRE O MÁXIMO DO MICRO EM 30 SEMANAS

PUBLICAÇÃO SEMANAL ILUSTRADA

Cr$ 8.200

13



Plotters: do papel para o vídeo

Cérebro artificial • Registre em PASCAL

rioGráfica

MANAUS, ALTAMIRA, MACAPÁ, JI.PARANÁ, PORTO VELHO, RIO BRANCO, SANTARÉM, BOA VISTA, SINOP, ALTA FLORESTA: Cr$ 10.660

MICROCOMPUTADOR CURSO PRÁTICO

APLICAÇÕES

O cérebro artificial 257

LINGUAGENS

Registre 260

PERIFÉRICOS

Traços e dígitos 263

RAIO X

Compaq Plus 266

PROJETOS DE PROGRAMAS

Códigos de comando 268

A revolução cubista 270

CIÊNCIA DA COMPUTAÇÃO

Métodos alternativos 272

TÉCNICAS DE PROGRAMAÇÃO

Biblioteca circulante 274

PERFIL

Eletrônica popular 276

TESTE

Depois de ler com atenção todo o fascículo, responda a estas perguntas. Veja a solução no próximo número, junto com um novo teste.

1) A inteligência artificial tem se concentrado em desenvolver sistemas que organizam o conhecimento humano em certas áreas bem delimitadas. Qual o nome desses sistemas?
2) Por que o Compaq Plus é relativamente lento no acesso a dados?
3) Qual a função do comando CMD no Commodore?
4) Qual o elemento-chave do traçador digital, localizado em suas articulações, que transforma os movimentos do braço em sinais elétricos?

RESPOSTAS DO TESTE ANTERIOR

1) A ciência que estuda os sistemas de controle e comunicação dos seres vivos e das máquinas é a cibernética, sistematizada por Norbert Wiener em 1947.
2) Antes de fazer qualquer reparo num micro deve-se desligá-lo da tomada.
3) C é a linguagem mais usada atualmente, pelas grandes software houses, para o desenvolvimento de programas.
4) Os computadores Lisa e Macintosh usam o processador de 32 bits 68000, da Motorola.

PLANO DA OBRA

MICROCOMPUTADOR - CURSO PRÁTICO é uma coleção de 30 fascículos de periodicidade semanal. Os 29 primeiros terão, cada um, vinte páginas internas (miolo) e quatro capas, além de uma introdução de dezesseis páginas acompanhando o fascículo 1. O fascículo 30 incluirá o índice geral da obra. Os miolos, encadernados, constituem dois volumes de um curso prático de microcomputação, com seções indicadas por tarjas de diferentes cores.
As duas primeiras capas são descartáveis; as 3.as e 4.as trazem programas de jogos. Encadernadas, formam um volume de cem páginas.

COMO ENCADERNAR

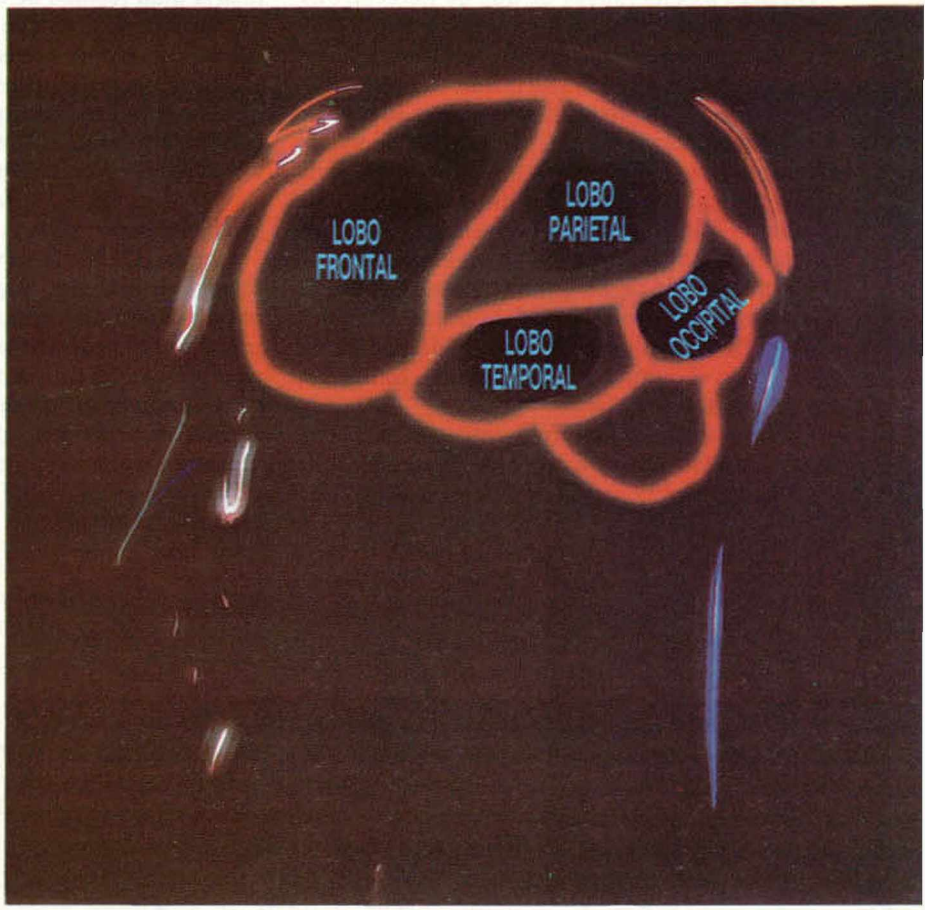
As capas duras, incluindo as guardas, estarão à venda simultaneamente com os fascículos 10 (**Volume 1**) e 30 (**Volume 2** e **Jogos**).
Volume 1 - Deve ser encadernado com os elementos dispostos nesta ordem: guardas; frontispício e introdução (publicados com o fascículo 1); miolos dos fascículos de 1 a 15; e guardas.
Volume 2 - Guardas; frontispício (que virá com o fascículo 30); miolos dos fascículos de 16 a 30; e guardas.
Jogos - Guardas; frontispício e sumário (que constituem as 3.as e 4.as capas do fascículo 30); 3.as e 4.as capas dos fascículos de 1 a 29; e guardas.

rioGráfica **Diretoria Executiva:** Oscar Neves; Nilo Sérgio de Almeida; Danilo Esteves Costa; **GRUPO EDUCAÇÃO E CULTURA - Planejamento e Comercialização - Gerente de Planejamento:** João Azeredo Coutinho; **Gerente de Marketing:** Jaime Rodrigues; **Gerente de Produto:** José Mauro Porto; **Gerente do P.C.P.:** Ayilton Menezes; **Diretor de Vendas:** Marcos March; **Gerente de Vendas:** Rubens Barbosa; **Coordenadores de Promoção:** Paulo Cesar M. Seixas; Edgar Mello M. Neto; **Gerente de Circulação:** Norberto Martin; **Supervisor de Operações:** Abel Pereira Pinto; **Editorial - Diretor:** João J. Noro; **Editor-chefe:** Maurício Rittner; **Administração Editorial:** Heitor de Souza Paixão; **Editor:** Jesse Navarro; **Revisores:** Isis Augusta Loyolla; Cacilda Guerra; Luiz Vicente Vieira Filho; Marcos Domingos Agathão; Maria da Graça Mendonça Couto; **Diretor de Arte:** Adalberto Cornavaca; **Chefe de Arte:** Bonifácio Duardes Miranda; **Chefe de Estúdio:** José Yuji Kuribayashi; **Assistente de Arte:** Miguel Luis Escámez Simón; **Colaboradores - Editores-assistentes:** Claudio Marcondes; Isa Mara Lando; Luiz Carlos Pizarro Marin; **Texto:** Paulo Brito; **Preparação de Texto:** Luiz Carlos Cardoso; **Tradução:** Fernando Dorfman Knijnik; Lúcia Maria Leal Gonçalves; Maria Clara Cescato; Maria Ester Menezes Martinho; Newton Roberval Eichemberg; Patrícia de Paiva e Castro; Regina Amarante; **Pesquisa:** Stella Maria Quentel; **Arte:** Edemilson Ribeiro da Silva; **Assessoria Técnica:** Geraldo Coen; Gustavo José Ferreira Grubba; Lauro de Lauro; Pierluigi Piazzi; Sérgio Jorge Batarce; Imarés Microcomputadores. As fotos não creditadas pertencem à obra original. Seguem-se os créditos das demais: Fernando Scavone; Microdigital. © APSIF Copenhagen, 1984; © Orbis Publishing Ltd., 1984; © Editora Rio Gráfica Ltda., 1985, para a língua portuguesa. **Composição:** Editora Rio Gráfica Ltda. e AM Produções Gráficas Ltda. **Impressão:** JBIG.

NÚMEROS ATRASADOS - Você poderá comprar os exemplares que faltam em sua coleção, pelo preço do último fascículo posto à venda, em sua banca de jornal preferida. Caso não consiga obtê-los, faça sua solicitação por carta endereçada diretamente à Editora Rio Gráfica Ltda., rua Itapiru, 1209, Rio Comprido, Rio de Janeiro, CEP 20251. O atendimento será feito por via postal. Nas cidades do Rio de Janeiro e de São Paulo, as compras poderão ser efetuadas pessoalmente nos seguintes endereços: Rio de Janeiro - Rua Itapiru, 1209, Rio Comprido; São Paulo - Rua Frei Caneca, 1152, Consolação.

Distribuidor exclusivo para todo o Brasil: Fernando Chinaglia - Rua Teodoro da Silva, 907, Rio de Janeiro. Distribuidor para Portugal: Electroliber Lda. - Rua Prof. Reinaldo dos Santos, 1488, Lisboa.

O CÉREBRO ARTIFICIAL



Mentes mecânicas
As pesquisas de inteligência artificial visam ao projeto de sistemas capazes de realizar tarefas que sempre exigiram a inteligência humana. No entanto, seu campo ampliou-se e passou a incluir funções perceptivas, como a visão e a audição. O objetivo é a programação de máquinas que simulem aspectos do comportamento e do raciocínio humanos.

As pesquisas sobre inteligência artificial são o que de mais avançado há no campo da informática. Nesta série de artigos, apresentaremos suas técnicas básicas, que você poderá explorar com o BASIC de seu micro.

Desde a metade da década de 70, vem aumentando o prestígio das pesquisas no campo da inteligência artificial, que passaram a ser realizadas inclusive por empresas particulares. Para entendermos a importância dessas pesquisas e os rumos que tomarão no futuro, nada melhor do que considerar seu passado.

Uma história resumida dessa área da ciência da computação poderia ser dividida em quatro partes, correspondentes a décadas e temas diferentes. Essa é, sem dúvida, uma simplificação grosseira, mas deixará evidentes os pontos mais importantes do assunto. Os temas são as respostas que um pesquisador de cada década daria à pergunta: De que trata a inteligência artificial?

Década de 50: Rede de neurônios
Década de 60: Pesquisa heurística
Década de 70: Sistemas especialistas
Década de 80: Aprendizagem por máquinas

Em 1943, Warren McCulloch e Walter Pitts desenvolveram um modelo matemático do neurônio do cérebro humano e animal, que serviu de base para uma representação simbólica da atividade cerebral. A partir dessa e de idéias semelhantes, outros estudiosos — entre os quais Norbert Wiener — elaboraram o que veio a ser chamado de "cibernética", de onde se originariam, na década de 50, as pesquisas de inteligência artificial.

A formalização do neurônio por McCulloch constituiu o fundamento das primeiras pesquisas de inteligência artificial. Dada a enorme complexidade do cérebro, não surpreende que tenha fracassado a tentativa de criar sistemas inteligentes a partir desse modelo.

Como o cérebro soluciona problemas de modo inteligente, o que pareceu mais promissor foi partir para a simulação de seu funcionamento. No entanto, nem o hardware nem o software da época estavam à altura dessa tarefa.

Dos sistemas então desenvolvidos, um dos poucos que tiveram algum sucesso foi o Perceptron, de Rosenblatt. Tratava-se de um sistema capaz de reconhecer padrões visuais predeterminados. Como mostra o diagrama da página 259, o Perceptron consiste numa rede de células fotossensíveis, simulando uma retina ocular em miniatura. Além disso, possui dispositivos detectores de subpadrões visuais predefinidos — chamados "demons" (demônios) — que monitoram (verificam ininterruptamente) o estado de grupos de células na rede. Ao reagirem à presença de subpadrões característicos, eles emitem um sinal para um centro de decisões. Este multiplica cada sinal enviado por um "demônio" por um fator de ponderação, positivo ou negativo, e soma os números resultantes. Se o total ultrapassa um limiar preestabelecido, o Perceptron indica "sim" (ou seja, há equivalência das imagens); caso contrário, diz "não". Assim, o Perceptron distingue entre duas imagens, embora seja possível, aplicando os mesmos princípios, fazer com que diferencie entre mais de duas.

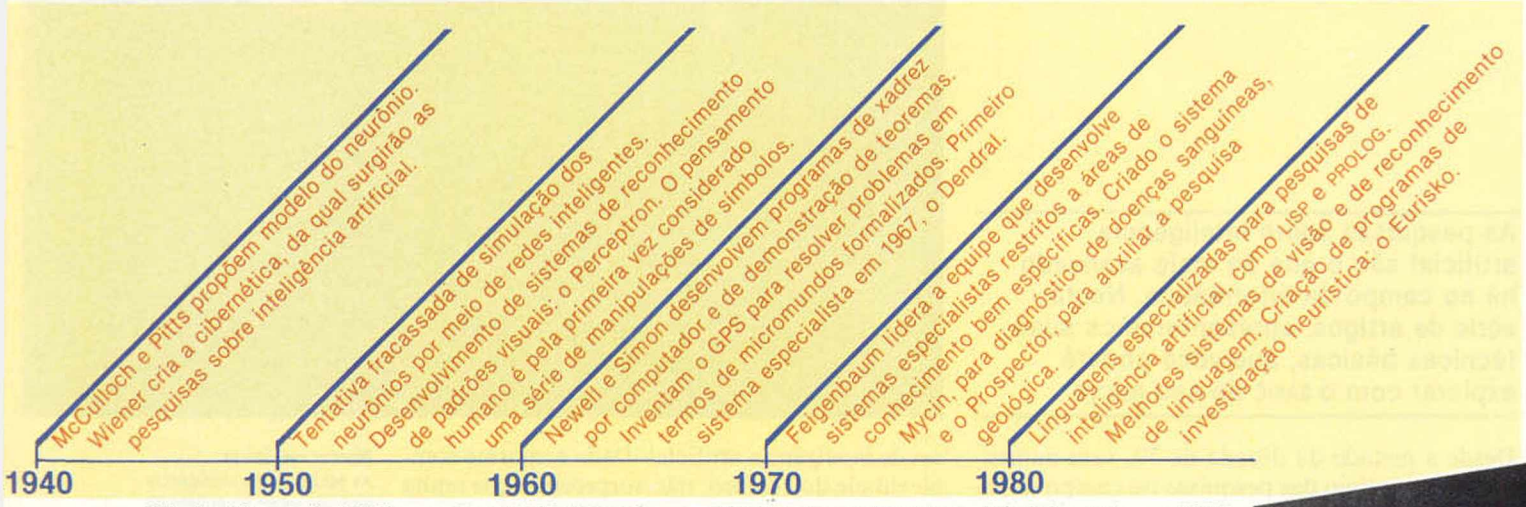
Quando as esperanças de que os Perceptrons pudessem resolver grande número de problemas se revelaram infundadas, os pesquisadores de inteligência artificial reconsideraram sua representação do pensamento humano, passando a vê-lo como coordenação de tarefas simples de manipulação de símbolos. Embora implicando mudança radical na orientação das pesquisas, isso significou que os projetos começaram a ser desenvolvidos sobre bases mais firmes. Afinal, os computadores já desempenhavam tarefas como a pesquisa e comparação de símbolos, consideradas o fundamento da resolução inteligente de problemas. A grande dificuldade era juntar essas atividades simples.

Os pesquisadores mais influentes da década de 60 foram Alan Newell e Herbert Simon, da Universidade Carnegie-Mellon, que trabalhavam com sistemas para a demonstração de teoremas e com jogos de xadrez por computador, entre outras investigações. A contribuição mais importante dos dois foi o programa Solucionador Geral de Problemas (em inglês, General Problem Solver — GPS). Era geral porque o usuário definia um "ambiente de tarefas", de acordo com os objetos de determinada área de estudos e com os operadores aplicáveis a esses objetos. No entanto, a generalidade limitava-se a problemas envolvendo um conjunto relativamente pequeno de estados e regras bem definidas. Por exemplo, problemas como o da Torre de Hanói, os de criptoaritmética (aritmética cifrada) e outros semelhantes, nos quais micromundos formalizados representam os parâmetros segundo os quais os problemas podem ser solucionados. O que o GPS não conseguia fazer era solucionar problemas da vida real, como elaborar um diagnóstico médico ou tomar decisões financeiras com base em moedas de valor flutuante.

O GPS baseava-se na idéia de que a solução de qualquer problema implica uma pesquisa no espaço de suas possíveis soluções. Para torná-la eficaz, era necessário orientar tal pesquisa por regras heurísticas, ou seja, regras que permitem o aprendizado a partir de descobertas próprias, conduzindo à solução desejada por um processo de tentativa e erro. Assim, um autômato utilizaria uma técnica de busca exaustiva para sair de um labirinto se não conhecesse sua estrutura; mas, se tivesse meios de saber quando estava próximo da saída, provavelmente atingiria seu objetivo com maior rapidez (embora, às vezes, o método heurístico conduza a becos sem saída). Nesse período, várias estratégias de pesquisa heurística foram desenvolvidas pelos estudiosos da inteligência artificial.

Como o GPS não servia para a solução de problemas da vida real, uma equipe liderada por Edward Feigenbaum, da Universidade de Stanford, procurou remediar essa falha. Em vez de tentar simular por computadores a inteligência genérica, eles se concentraram, já na década de 70, em áreas bem específicas do conhecimento, desenvolvendo os sistemas especialistas.

Perspectiva histórica
As pesquisas de inteligência artificial vêm se tornando cada vez mais importantes em termos práticos.
No entanto, sua história é curta em relação à de outros campos de pesquisa, como se pode ver no diagrama.



O primeiro, o Dendral, foi elaborado em 1967 e analisava espectogramas de massa. No entanto, o mais influente foi o Mycin, apresentado em 1974, que diagnosticava infecções bacteriológicas do sangue e prescrevia drogas para o tratamento. O Mycin deu origem a toda uma família de sistemas de diagnóstico médico, alguns já em uso rotineiro, como o Puff, capaz de diagnosticar funções pulmonares, instalado no Pacific Medical Center, nos Estados Unidos.

Várias dentre as características introduzidas pelo Mycin foram adotadas por todos os sistemas especialistas. Em primeiro lugar, o "conhecimento" deles consiste em centenas de regras como a seguinte:

Regra n.º 47

SE
1) o local da cultura é o sangue, e
2) não se conhece ao certo a identidade do organismo, e
3) a coloração do organismo é gram-negativa, e
4) a morfologia do organismo é a de bastonete, e
5) o paciente sofreu queimaduras graves,

ENTÃO há uma pequena evidência (0,4) sugerindo que o organismo seja pseudomonas.

Edward Feigenbaum
Como chefe de uma equipe da Universidade de Stanford, na Califórnia, Feigenbaum criou os primeiros sistemas especialistas. Isso implicou uma reviravolta nas pesquisas, que passaram dos programas de inteligência genérica aos sistemas com objetivos e conhecimentos estritamente definidos.

Em segundo lugar, essas regras são probabilísticas. O criador do Mycin, um médico chamado Shortliffe, elaborou um esquema baseado em fatores de certeza, permitindo ao sistema tirar conclusões plausíveis a partir de dados incertos. Desse modo, o número 0,4 não é, a rigor, uma probabilidade, mas um "fator de significância". O Mycin e outros sistemas do mesmo tipo chegam a conclusões corretas mesmo partindo de informações incompletas e parcialmente errôneas. Eles conseguem esses resultados por meio de métodos de raciocínio aproximativo, como a lógica da incerteza, o cálculo de probabilidades, o recurso dos fatores de certeza etc.

Uma terceira característica é a capacidade do Mycin para explicitar seu próprio processo de raciocínio. O médico pode colocar-lhe vários tipos de pergunta com o objetivo de saber como ele chegou a uma conclusão específica ou por que requisitou determinada informação.

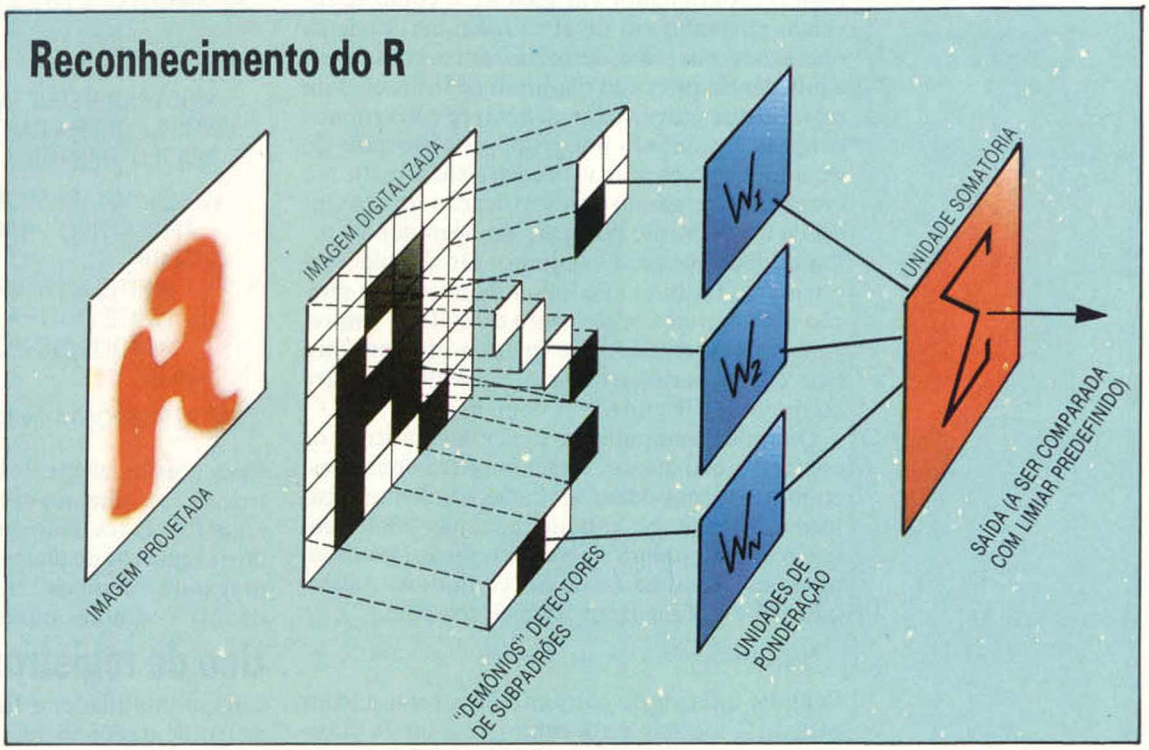
Por fim, o que é mais importante, o Mycin funciona. Ele faz o que requer anos de treino de um ser humano. Na verdade, o Mycin é mais empregado no ensino do que em diagnósticos, mas as grandes empresas, os órgãos governamentais e a mídia já começam a se interessar pelos princípios que ele demonstra.

Na década de 80, os sistemas especialistas ficaram na moda, graças a seu "ingrediente ativo" — o conhecimento. E a qualidade e abrangência de sua base de conhecimentos é diretamente responsável pelo sucesso de um sistema especialista. Mas o conhecimento não é algo que se possa introduzir com facilidade num computador. Codificar as habilidades de um especialista humano é um processo longo e trabalhoso. Assim, enquanto o mundo se maravilha com os sistemas especialistas, os pesquisadores da inteligência artificial dedicam-se a outro problema: o do aprendizado das máquinas, ou seja, os métodos de aquisição automática de conhecimentos. O campo da inteligência artificial sempre foi muito fluido e, no início de 1985, as pesquisas mais avançadas concentravam-se no Eurisko.

O Eurisko é um programa de investigações que amplia e melhora automaticamente, por indução, seu corpo de regras heurísticas. Vem sendo aplicado com sucesso a vários problemas práticos. Um dos resultados, na área de projetos de circuitos integrados, foi a invenção de uma inovadora porta lógica de três dimensões. Sistemas como o Eurisko constituem as pesquisas mais avançadas na área da inteligência artificial. Como esta constitui a vanguarda da ciência da computação, o que está em jogo é, nada mais nada menos, o futuro da informática.

A grande ironia na evolução resumida aqui é que, ao concentrar-se nos problemas de aprendizado, os estudiosos da inteligência artificial retornaram ao ponto de partida, pois a questão-chave nos primeiros tempos da cibernética era justamente a do aprendizado.

Nesta série de artigos sobre as áreas mais problemáticas da inteligência artificial, todos os exemplos de programas serão apresentados em BASIC. Embora a maioria das pesquisas em linguagem artificial seja realizada em LISP e PROLOG, o BASIC de seu micro servirá muito bem para a exploração inicial de algumas técnicas importantes. Afinal, um micro atual é tão poderoso quanto um computador de grande porte de vinte anos atrás. Além disso, uma vez entendidos os métodos básicos, você será capaz de implementá-los em qualquer outra linguagem.



Percepções do Perceptron
A imagem a ser analisada (a letra R) é projetada num plano e digitalizada. Os "demônios" examinam um grupo de pixels (por exemplo, quatro de cada vez) e indicam a presença ou ausência do padrão que foram programados para reconhecer. Cada resposta (expressa em dígitos zero e um) é multiplicada por um fator de ponderação, de acordo com sua importância dentro do padrão completo, e então somam-se as respostas ponderadas. O resultado é comparado a um valor limite. Se atingir o limiar, o sistema indica que o padrão é igual à forma original.

LÓGICA DA INCERTEZA

Lógica que procura reproduzir a habilidade humana de lidar com a incerteza e as meias-verdades e de tomar decisões com base em dados incompletos. Para tanto, apóia-se em valores múltiplos, possibilidades e probabilidades.

REGISTRE

Os registros são o método mais prático, no PASCAL, para agrupar diferentes tipos de dados. Antes de examinar essa nova estrutura, porém, veremos algumas operações com conjuntos.

Esta série já mostrou a utilidade dos conjuntos para o programador em PASCAL e como se revelam eficientes em nível de máquina, onde as operações por eles definidas encontram uma equivalência direta no conjunto de instruções da maioria dos processadores. Sempre construímos conjuntos reunindo um grupo de elementos de qualquer tipo escalar verdadeiro (não real). No entanto, excetuando-se a verificação da existência de um elemento por meio do operador IN, todas as operações com conjuntos são definidas na estrutura. Embora não haja mecanismo de seleção para extrair um elemento específico, empregamos os operadores de inclusão de conjunto (<= e >=) para verificar a equivalência entre conjuntos ("igualdade") com = e < >.

Quando manipulados por matemáticos, os conjuntos não apenas contêm objetos ilimitados, como também podem ser considerados teoricamente infinitos. No entanto, qualquer implementação exigirá algum tipo de limitação, tanto no tamanho como na faixa dos conjuntos. Assim, não tente definir tipos ambiciosos como:

```
NATURAIS = SET OF INTEGERS;
```

O limite inferior do conjunto deve ter um valor ordinal — de zero para cima — e o limite máximo estará restrito a algum valor absoluto — entre 255 e 4.095, nas implementações em micros. Observe que o conjunto vazio (representado literalmente por []) é membro de todos os conjuntos possíveis — qualquer que seja o tipo. Isso parece uma brecha na estrutura fortemente tipificada do PASCAL, mas, na prática, é possível deduzir o tipo do conjunto vazio a partir do tipo de outros conjuntos em qualquer expressão.

A inclusão estrita de subconjuntos verdadeiros não é direta, e tampouco se definem os operadores < e > para estruturas de conjuntos. Isso se deve a razões de implementação — a maioria das restrições do PASCAL justifica-se do ponto de vista da eficácia ou da lógica pura. Para testar a inclusão estrita é necessário um teste duplo, por exemplo:

```
(A >= B) AND (A <> B)
```

O teste para FEITO no programa de Bingo não exigia essa verificação especial, pois o conjunto dos números chamados é equivalente ("igual") ou, o mais provável, um sobreconjunto dos números na cartela. Desse modo, a expressão booleana seria expressa como

```
FEITO := CHAMADOS >= CARTELA
```

que será verdadeira quando todos os membros de CARTELA estiverem contidos no conjunto CHAMADOS. Esse tipo de propriedade faz dos conjuntos uma importante estrutura de dados para a solução de problemas. Por enquanto, o emprego mais produtivo dos conjuntos consiste no teste de subconjuntos do tipo CHAR. O esquema abaixo (em PASCAL/português) facilita a programação de qualquer software interativo:

```
NEGATIVO := ['N', 'n'];
AFIRMATIVO := ['S', 's'];

REPEAT
  {JOGAR O JOGO}
  {MOSTRAR RESULTADO}
  WRITE ('OUTRA PARTIDA?');
  READLN (RESPOSTA);

  WHILE NOT (RESPOSTA IN
    AFIRMATIVO + NEGATIVO) DO
    BEGIN
      WRITELN ('RESPONDA S(IM) OU N(AO)');
      WRITE ('OUTRA PARTIDA?');
      READLN (RESPOSTA);
    END;

UNTIL RESPOSTA IN NEGATIVO
```

Para acessar elementos individuais de uma estrutura, escolhemos entre matrizes, arquivos ou registros. Estes possuem a capacidade de reproduzir registros de dados usados na vida real, empregando "campos" mistos de qualquer tipo de dados — simples ou estruturados.

Uso de registros pelo PASCAL

Em contabilidade, a forma mais comum de registro de dados inclui campos para nomes, endereços, números de telefone, códigos de contas etc. O importante é manipular esses dados como um bloco de informações, bem como acessar qualquer campo individual e processar o dado de modo adequado. O PASCAL possibilita a atribuição (e manipulação por outros métodos) de tais objetos como uma unidade, mas também permite o acesso a qualquer campo, para comparação ou processamento, de acordo com o tipo específico de dado. A definição de um campo misto é bastante simples:

```
TYPE
  SALA = RECORD
           NUMERO : 1 .. 999;
           POSICAO : (NORTE, LESTE, SUL,
             OESTE);
           OCUPADA : BOOLEAN
         END; {SALA}
VAR
  ESCRITORIO : SALA;
```

Assim como a instrução CASE constituía uma exceção ao uso da palavra reservada END como delimitador, a definição de um registro é a única exceção na parte declarativa à regra de que as palavras BEGIN e END devem ser utilizadas em pares. Por esse motivo, convém comentar o END de um registro com seu identificador. Qualquer variável do tipo SALA compõe-se de três campos. No exemplo, cada campo é de um tipo escalar diferente, mas eles também poderiam ser do mesmo tipo — simples ou estruturado. Não há qualquer restrição aos tipos admitidos no interior de campos de registros. Admite-se até mesmo um campo formado por um conjunto de arquivos de registros contendo conjuntos!

Entre as palavras delimitadoras RECORD e END, a sintaxe de definição é exatamente a mesma relativa à declaração VAR. Aqui, no entanto, estamos declarando identificadores de campo que fazem parte da estrutura do registro. Assim, os nomes NUMERO, POSICAO e OCUPADA não existem fora do "âmbito" do identificador do registro. Esses nomes de campo são identificadores locais e poderiam duplicar os nomes de variáveis no programa. Eles são acessados por meio dos dois mecanismos de seleção para registros do PASCAL: a notação de "ponto" e a instrução WITH.

Para selecionar determinado campo com o uso do ponto, todo o identificador do registro é separado do identificador de campo que vem a seguir por meio de um ponto ("."). Por exemplo,

```
ESCRITORIO.NUMERO
```

só se referiria ao campo de subfaixa integer. Poderíamos inicializar o conteúdo do registro com as instruções

```
READ (ESCRITORIO.NUMERO);
ESCRITORIO.POSICAO := LESTE;
ESCRITORIO.OCUPADO := PESSOAL < > [];
```

e assim por diante. Observe que usamos o conjunto vazio com um tipo que depende de PESSOAL, porém sem declará-lo.

A instrução WITH

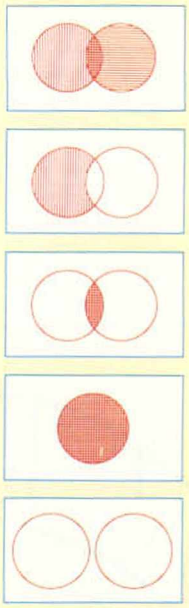
A notação de ponto torna-se inconveniente quando queremos acessar a maioria ou todos os campos de um registro. Existe uma instrução estruturada alternativa que indica os identificadores de campo. A semântica da instrução WITH significa, de modo aproximado: "Para fazer algo com esse registro, preciso especificar somente os nomes de campo". A sintaxe da instrução WITH possui a mesma forma do loop WHILE, e a seqüência de atribuições de inicialização seria melhor expressa por:

```
WITH ESCRITORIO DO
  BEGIN
    NUMERO := 123;
    POSICAO := OESTE;
    OCUPADA := TRUE
  END
```

Operações com conjuntos no PASCAL

Estão relacionadas aqui todas as operações com conjuntos existentes no PASCAL (com os correspondentes diagramas de Venn, quando apropriado).

1. União (S1 + S2): sobreconjunto que abrange todos os objetos, tanto do conjunto S1 como do S2 (diagrama 1).
2. Diferença (S1 – S2): subconjunto dos membros de S1 que também não pertencem a S2 (diagrama 2).
3. Interseção (S1 * S2): subconjunto de objetos comuns a ambos os conjuntos (diagrama 3).
4. Equivalência (S1 = S2): quando os membros de S1 e S2 são idênticos (diagrama 4).
5. Não equivalência (S1 < > S2): verdadeira se nenhum dos membros de S1 for membro de S2 e vice-versa (diagrama 5).
6. Inclusão (S1 < = S2): verdadeira se todos os membros de S1 também forem membros de S2.
7. Inclusão (S1 > = S2): verdadeira se todos os membros de S2 também forem membros de S1.
8. Pertinência (m IN S1): verdadeira se o conjunto de elemento único ([m]) for um subconjunto de S1.



Tacada perfeita

O problema apresentado na matéria anterior, no qual pedimos o cálculo da maior seqüência de tacadas num jogo de bilhar, resolve-se assim:

```
PROGRAM    BREAKMAX    (OUTPUT);

CONST
  VERMELHAS = 15;

TYPE
  SNOOKER = (BRANCA, VERMELHA, AMARELA, VERDE,
    MARROM, AZUL, ROSA, PRETA);

VAR
  BOLA1: SNOOKER;
  PONTOS: VERMELHAS .. MAXINT;

BEGIN
  PONTOS := VERMELHAS * (ORD(VERMELHA) + ORD(PRETA));
  FOR BOLA1 := AMARELA TO PRETA DO
    PONTOS := PONTOS + ORD(BOLA1);
  WRITELN ('O BREAK MAXIMO E:', PONTOS : 3);
END.
```

Em toda a instrução WITH (desde BEGIN até END, nesse caso), os identificadores de campo não precisam ser qualificados pelo identificador de registro e pelo ponto, e a referência a eles é direta. Não há confusão se outra variável do programa recebe o nome Numero, por exemplo. O âmbito local sempre tem prioridade. Todavia, usando a notação de ponto, comunicamos valores para além dos limites do âmbito. Por exemplo,

```
ESCRITORIO.NUMERO := Numero
```

atribuiria um valor da variável exterior (Numero) ao campo de registro (NUMERO). O PASCAL não distingue maiúsculas e minúsculas, e o identificador não qualificado NUMERO refere-se a uma variável externa, a menos que esteja numa instrução WITH. De qualquer forma, seria essencial utilizar a notação de ponto se tivéssemos uma atribuição entre duas variáveis do mesmo tipo de registro, tal como

```
ESPERA.POSICAO := RECEPCAO.POSICAO
```

A instrução WITH é usada apenas para indicar um dos campos de variável, caso contrário haveria uma ambigüidade inaceitável para o compilador. O melhor que poderíamos fazer seria:

```
WITH ESPERA DO
  POSICAO := RECEPCAO.POSICAO
```

Portanto, as duas formas de notação têm sua utilidade. Em geral, a forma mais adequada torna-se evidente durante a própria utilização.

Corrida de longa distância

O programa Distâncias lê dois comprimentos, expressos em jardas, pés e polegadas, com o uso do teclado. Definimos um registro a fim de reservar um campo separado para cada valor das diferentes unidades de medida. O programa soma os comprimentos, transportando as polegadas para o campo "POL" e os pés excedentes para o campo "PES", usando os operadores integer DIV e MOD para obter a soma e o resto. Os resultados são atribuídos aos campos de "TOTAL" e depois impressos. (A conversão dessas unidades de medida para o sistema métrico decimal seria a seguinte:

1 jarda = 0,9144 metro
1 pé = 0,3048 metro
1 polegada = 0,0254 metro)

```
PROGRAM     DISTANCIAS     (INPUT, OUTPUT);
CONST
  BYTEMAXIMO = 255;

TYPE
  BYTE              = 0 .. BYTEMAXIMO;

  DISTANCIA         = RECORD
                        JAR     : 0 .. 11;
                        PES     : 1 .. 2;
                        POL     : BYTE;

                      END  DISTANCIA;
VAR
  POLEGADA;
  PE                : BYTE;
  DISTANCIAA,
  DISTANCIAB,
  TOTAL             : DISTANCIA;

BEGIN

  WRITELN ('ENTRE DISTANCIAS COMO JARDAS, PES E
    POLEGADAS');
  WRITELN ('SEPARADOS POR ESPACO OU RETURN');
  WRITELN;
  WRITE ('PRIMEIRA :' : 15);
  READ (DISTANCIAA.JARDAS, DISTANCIAA.PES,
    DISTANCIAA.POL);
  WRITE ('SEGUNDA :' : 15);

  WITH DISTANCIAB DO
    READ (JARDAS, PES, POL);

  POLEGADA := DISTANCIAA.POL + DISTANCIAB.POL;
  PE := DISTANCIAA.PES + DISTANCIAB.PES + POLEGADA
    DIV 12;
  WRITELN;

  WITH TOTAL DO
    BEGIN
      POL := POLEGADA MOD 12;
      PES := PE MOD 3;
      JAR := DISTANCIAA.JAR + DISTANCIAB.JAR + PE
        DIV 3;
      WRITELN ('A DISTANCIA TOTAL E :');
      WRITELN (JAR : 25, 'JARDAS', PES : 1,
        'PE(S)', POL : 1, 'POL');
    END;
END.
```

Proteção dos dados

No programa Bingo, apresentado no artigo anterior, estabelecemos um exercício em que você devia acrescentar estruturas em loop ao programa para evitar a repetição de um número já chamado e detectar quaisquer números ilegítimos além da faixa de 1 a 90. O melhor modo de proteger o programa Bingo contra a entrada errônea de dados consiste em acrescentar uma estrutura WHILE após as duas instruções READLN. A estrutura será a mesma em cada caso:

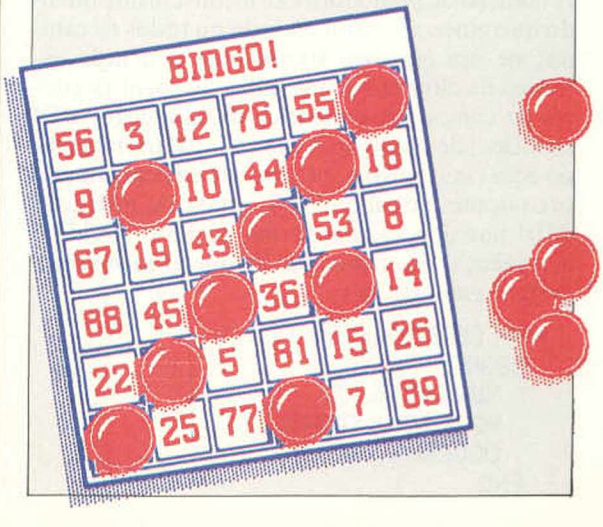
```
WHILE o número for ilegítimo DO
  fornecer uma mensagem de erro adequada
  escrever um outro indicador
  ler os dados novamente
```

No caso das entradas iniciais de dados, devemos nos assegurar de que o número fornecido não está fora da faixa permitida (de 1 a 90), bem como evitar a repetição de uma entrada anterior. Assim:

```
WHILE NOT (NUMERO IN POSSIVEIS)
  OR (NUMERO IN CARTELA) DO
  BEGIN
    WRITELN (NUMERO : 20, 'E INVALIDO');
    WRITE ('REENTRE :' : 14);
    READLN (NUMERO);
  END;
    {ETC.}
```

A mesma estratégia se aplica à chamada dos números, mas modifica-se a condição para a entrada do loop de validação para:

```
WHILE NOT (NUMERO IN POSSIVEIS)
  OR (NUMERO IN CHAMADOS) DO
  {ETC.}
```



TRAÇOS E DÍGITOS

Apesar de muitos micros possuírem excelentes recursos gráficos, a transferência de uma foto ou desenho do papel para o vídeo revela-se demorada e difícil. A melhor solução é o traçador digital.

Um traçador digital é um equipamento simples que reproduz as linhas de uma planta, fotografia ou desenho e, ao mesmo tempo, transfere-as para a tela do computador. Trata-se, portanto, de um periférico que simplifica bastante a introdução de imagens no computador. A facilidade do processo depende sobretudo do software aplicativo que acompanha o equipamento.

O princípio de funcionamento é basicamente o mesmo em todos os modelos de traçadores digitais. Um ponteiro na extremidade de um braço articulado envia sinais elétricos ao computador. Estes variam em magnitude conforme a posição do ponteiro sobre a figura, sendo depois convertidos em sinais digitais pelo computador e usados para desenhar um ponto na tela, no lugar correto.

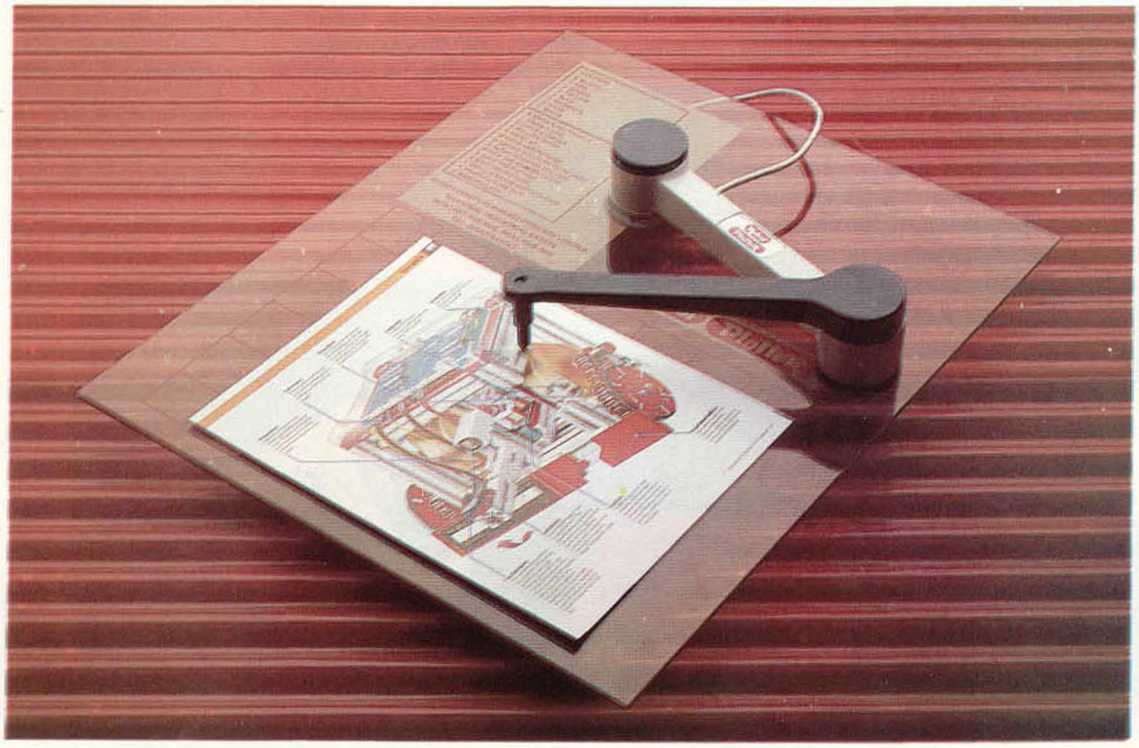
Todos os traçadores são comercializados com um programa aplicativo que executa essa tarefa e que dispõe de vários recursos — por exemplo, desenhar linhas em diversas cores. Outros aplicativos permitem selecionar diferentes modos de apresentação na tela, diminuindo a resolução à medida que se aumenta o número de cores, em função do limite de espaço disponível na memória.

Um traçador digital que vem obtendo êxito no mercado internacional é o Robot Plotter. Produzido pela empresa inglesa Robot Computer Development, vem montado numa prancheta transparente, onde um quadriculado representa as posições dos pixels. O braço do traçador — fixado numa extremidade da prancheta e feito em metal e plástico — é muito resistente. O ponteiro assemelha-se a uma caneta que se projeta do braço.

A figura a ser copiada fica embaixo da base transparente. Esse sistema, contudo, dificulta a visualização da figura à medida que ela vai sendo copiada.

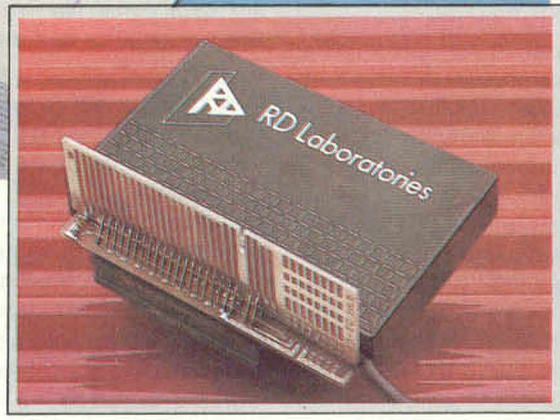
Acompanha o Robot Plotter uma fita cassete que traz o programa aplicativo. Além das rotinas de traçado, o programa contém várias outras para desenhar círculos, retângulos e linhas.

Programas traçadores desse tipo armazenam todas as imagens como uma série de linhas; assim, o desenho de um mapa, por exemplo, é introduzido na memória como uma seqüência de linhas curtas. Isso facilita a remoção das linhas indesejáveis sem afetar outras, contíguas. Uma imagem complexa requer muito espaço de memória — às vezes, mais do que o micro dispõe.



Robot Plotter
A imagem a ser copiada é colocada sob a prancheta transparente. O software aplicativo inclui rotinas para o traçado de círculos e retângulos.

Traçador digital RD Labs
Esse modelo, de plástico e muito leve, é vendido sem a prancheta, mas pode ser utilizado em qualquer superfície, graças a seu suporte aderente.



Interfaces especiais

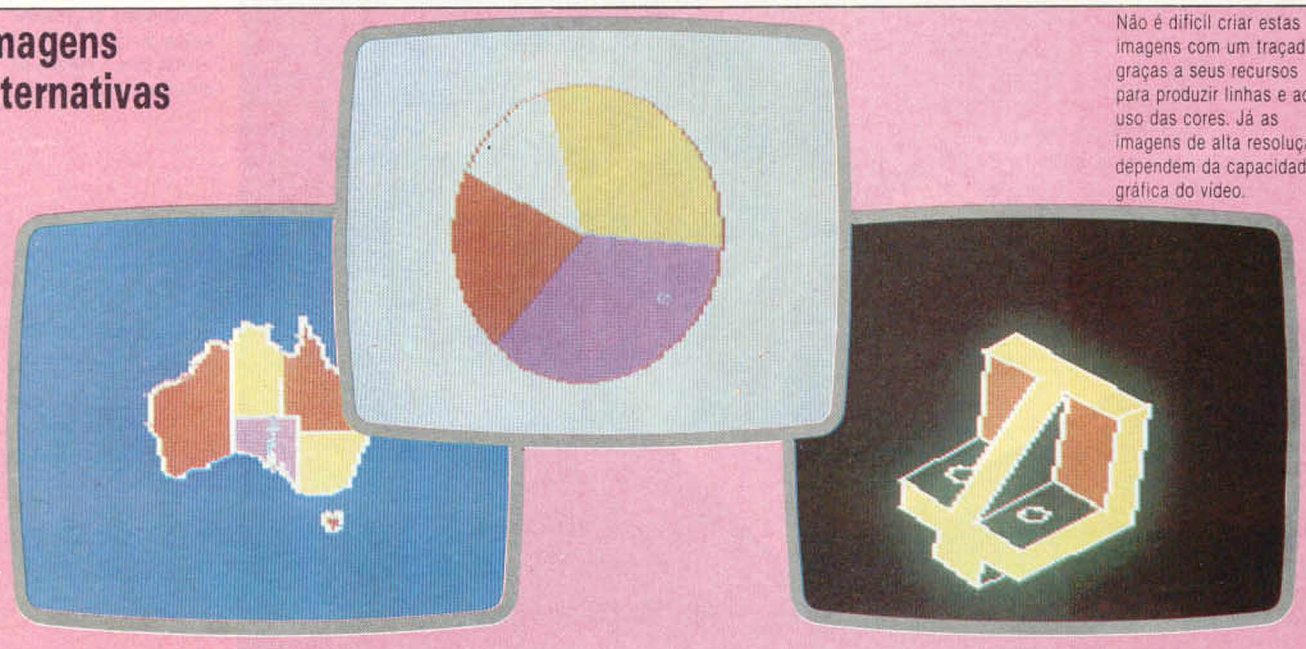
Os fabricantes do traçador digital RD Labs desenvolveram uma interface especial para a conexão de seu equipamento ao micro Spectrum.

No entanto, por serem armazenadas sob a forma de seqüências de linhas, é relativamente fácil transferir para outros programas as imagens criadas por meio do traçador.

O traçador Digigraph também é bastante resistente. A placa da base consiste numa grande prancheta de madeira com um quadriculado vermelho pintado na superfície. O braço, um tubo de alumínio, vem equipado com um disco transparente na extremidade do ponteiro. Coloca-se a figura a ser copiada sobre a prancheta; o braço move-se com suavidade e a transparência do disco permite visão clara da imagem.

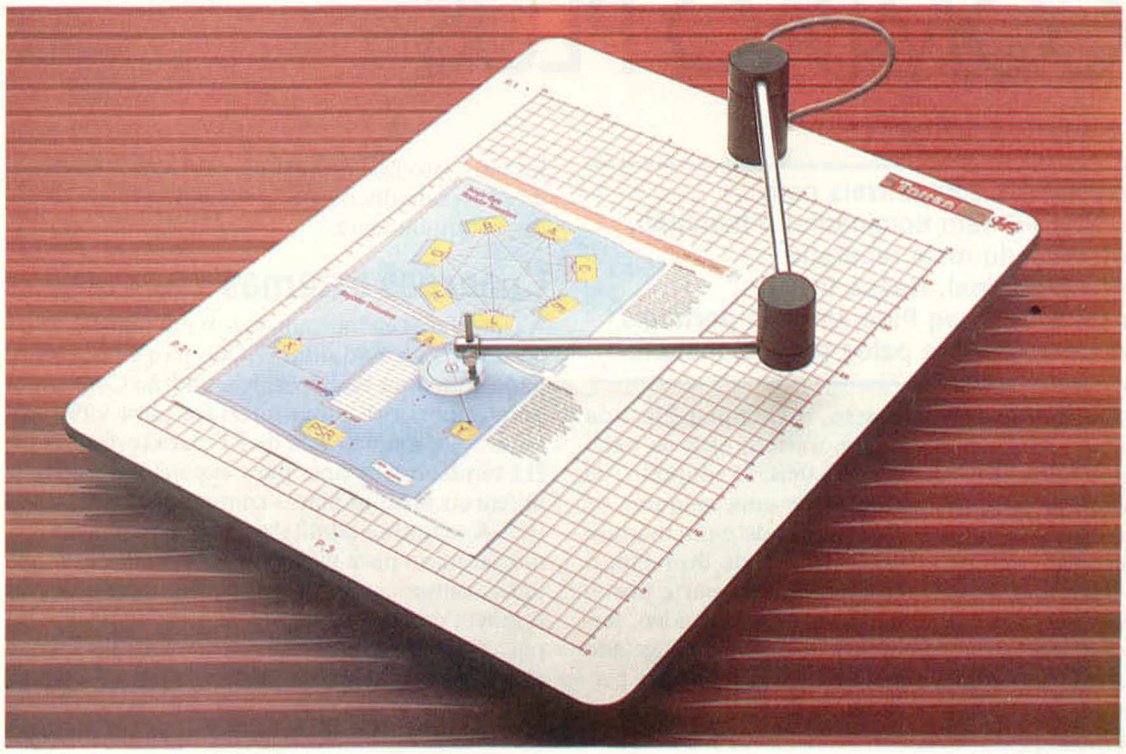
O aplicativo que acompanha o Digigraph, embora menos sofisticado que o do Robot Plotter, oferece recursos semelhantes. Os movimentos do traçador não se armazenam sob a forma de comandos isolados para o computador e, portanto, as figuras são transferidas diretamente para a tela. A gravação e a leitura das imagens se fazem a partir da memória de vídeo, o que torna mais difícil alterá-las. Todavia, esse processo per-

Imagens alternativas



Não é difícil criar estas imagens com um traçador, graças a seus recursos para produzir linhas e ao uso das cores. Já as imagens de alta resolução dependem da capacidade gráfica do vídeo.

Digigraph
Com braço de alumínio e plástico e prancheta de madeira, o traçador Digigraph é um dos mais resistentes do mercado.



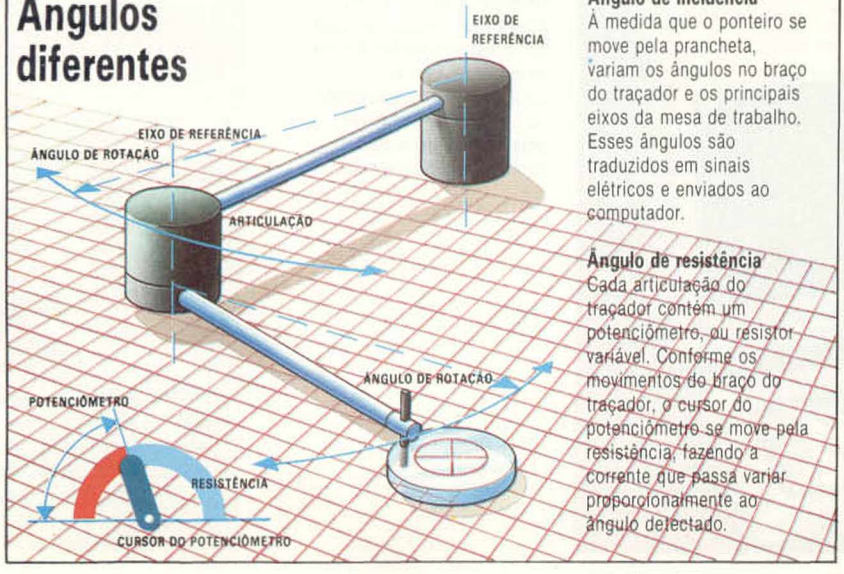
mite que a memória necessária para a elaboração de uma figura complexa não seja superior à exigida por uma figura simples.

O traçador digital RD Labs é comercializado em duas versões — uma para o BBC Micro e outra para o Spectrum, da Sinclair, cujo compatível brasileiro é o TK 90X da Microdigital. Nos dois modelos, o braço vem solto, com um suporte aderente que permite sua fixação à superfície apropriada. De plástico e muito flexível, o braço opera com razoável precisão. O ponteiro utiliza dois pequenos plásticos cruzados, o que atrapalha um pouco seu uso.

Na versão para o BBC Micro, as imagens são gravadas de forma semelhante à do Robot Plotter, havendo o mesmo problema de escassez de memória. O aplicativo em cassete, bastante complexo, contém rotinas para desenho de círculos, retângulos e linhas, além de um recurso para animação de imagens. A versão para o Spectrum é basicamente a mesma, mas conta com um conversor analógico-digital — inexistente no outro modelo.

Os traçadores digitais ainda não estão sendo fabricados no Brasil. Portanto, a única alternativa para o usuário que necessite de um equipamento desse tipo é a importação e adaptação de algum dos modelos existentes no exterior. Para adequá-lo a seu micro, serão necessárias duas providências trabalhosas: modificação do conector que faz a ligação do traçador com a saída para joystick ou para paddle; e compatibilização do software aplicativo com a versão BASIC de seu aparelho.

Ângulos diferentes



Ângulo de incidência
À medida que o ponteiro se move pela prancheta, variam os ângulos no braço do traçador e os principais eixos da mesa de trabalho. Esses ângulos são traduzidos em sinais elétricos e enviados ao computador.

Ângulo de resistência
Cada articulação do traçador contém um potenciômetro, ou resistor variável. Conforme os movimentos do braço do traçador, o cursor do potenciômetro se move pela resistência, fazendo a corrente que passa variar proporcionalmente ao ângulo detectado.

Registre

Linha Apple

TELEX

Este pequeno programa em BASIC permite o movimento automático das informações contidas na tela.

```
100  REM ***  TELEX PARA LINHA APPLE ***
110  HOME
120  PRINT "ENTRE COM UM STRING E ELE PA
     SSARA EM":PRINT "SCROLL PELA SUA TE
     LA DA ESQUERDA PARA"
130  PRINT "A DIREITA":INPUT A$
140  HOME:VTAB 11
150  X=LEN(A$):IF X>39 THEN 110
160  X=40
170  HTAB 1:PRINT LEFT$(A$,X);:A$=MID$(A
     $,2)+LEFT$(A$,1)
180  FOR S=1 TO 75:NEXT S
190  K=PEEK (-16384)
200  IF K<128 THEN K=FRE(0):GOTO 170
210  END
```

COMPAQ PLUS

Os micros compatíveis com o IBM PC vêm dominando o mercado, oferecendo mais recursos que o original. Nessa trilha está o Compaq Plus, um dos portáteis mais cobiçados pelos profissionais.

Apesar dos 14 kg de peso, o Compaq Plus, da Compaq Computer Corporation, está na categoria dos profissionais portáteis. O conjunto, de dimensões aproximadas às de uma máquina de costura, compõe-se de um teclado e um módulo central contendo duas unidades de disquete, o vídeo e as placas. Há uma alça na parte posterior, e o teclado, fechando-se contra o vídeo, serve como base do conjunto. A conexão do teclado à máquina é feita por cabo espiralado, o que permite afastá-lo por cerca de 20 cm; além disso, ambos possuem ajuste de inclinação, para posicionamento no ângulo mais confortável ao operador.

Com desenho idêntico ao do IBM PC, o teclado possui dez teclas de função e dez de calculadora, além das principais, tipo QWERTY. A tecla é um monitor de fósforo verde com dimensões de 17,7 x 13,3 cm, que mostra 80 x 25 caracteres com boa nitidez e controle de luminosidade.

O Compaq Plus incorpora duas unidades de discos flexíveis de 5 1/2 polegadas. Como a configuração padrão do equipamento é de apenas uma unidade de disco e o sistema operacional MS-DOS pede que todos os discos sejam nela colocados, isso pode constituir um incômodo quando se copiam discos, pois o usuário precisa trocá-los continuamente.

Conexões externas

As interfaces encontram-se sob um painel, no lado direito da máquina. O Compaq Plus é equipado com uma saída paralela padrão Centronics para impressora, uma interface para vídeo em cores RGB e uma saída de RF para tevê comum. Há também três slots para expansões, que permitem encaixar as placas compatíveis com o IBM PC. A máquina admite ainda expansões de memória, placa para vídeo em cores ou um modem para comunicação via rede telefônica. Dispõe também de um ventilador para refrigeração das placas. Os circuitos vêm com revestimento metálico (principal responsável pelo peso), destinado a blindar o equipamento contra interferências de rádio e a dissipar o calor, além de proteger de manuseio descuidado.

O Compaq Plus, tal como o IBM PC, utiliza um processador 8088 da Intel. Alguns de seus rivais — entre eles o M25 da Olivetti — usam um chip mais avançado, o 8086. Apesar de o 8088 ser um processador de 16 bits com um bus de endereçamento de 20 bits, possui um bus de dados de 8 bits, contra o de 16 bits do 8086. Assim, embora rode na mesma velocidade que o IBM PC, o Compaq é mais lento no acesso aos dados que seus competidores equipados com 8086. O ponto alto da máquina está em ser um dos mais confiáveis compatíveis ao IBM PC à disposição no mercado internacional.

Compacto
As linhas elegantes e harmoniosas do gabinete do Compaq refletem-se no painel frontal e no teclado, este muito semelhante ao do IBM PC. Aberturas em ambos os lados dão acesso às saídas, à fonte de alimentação e ao compartimento que acomoda o fio condutor de energia — detalhe importante que costuma ser negligenciado.

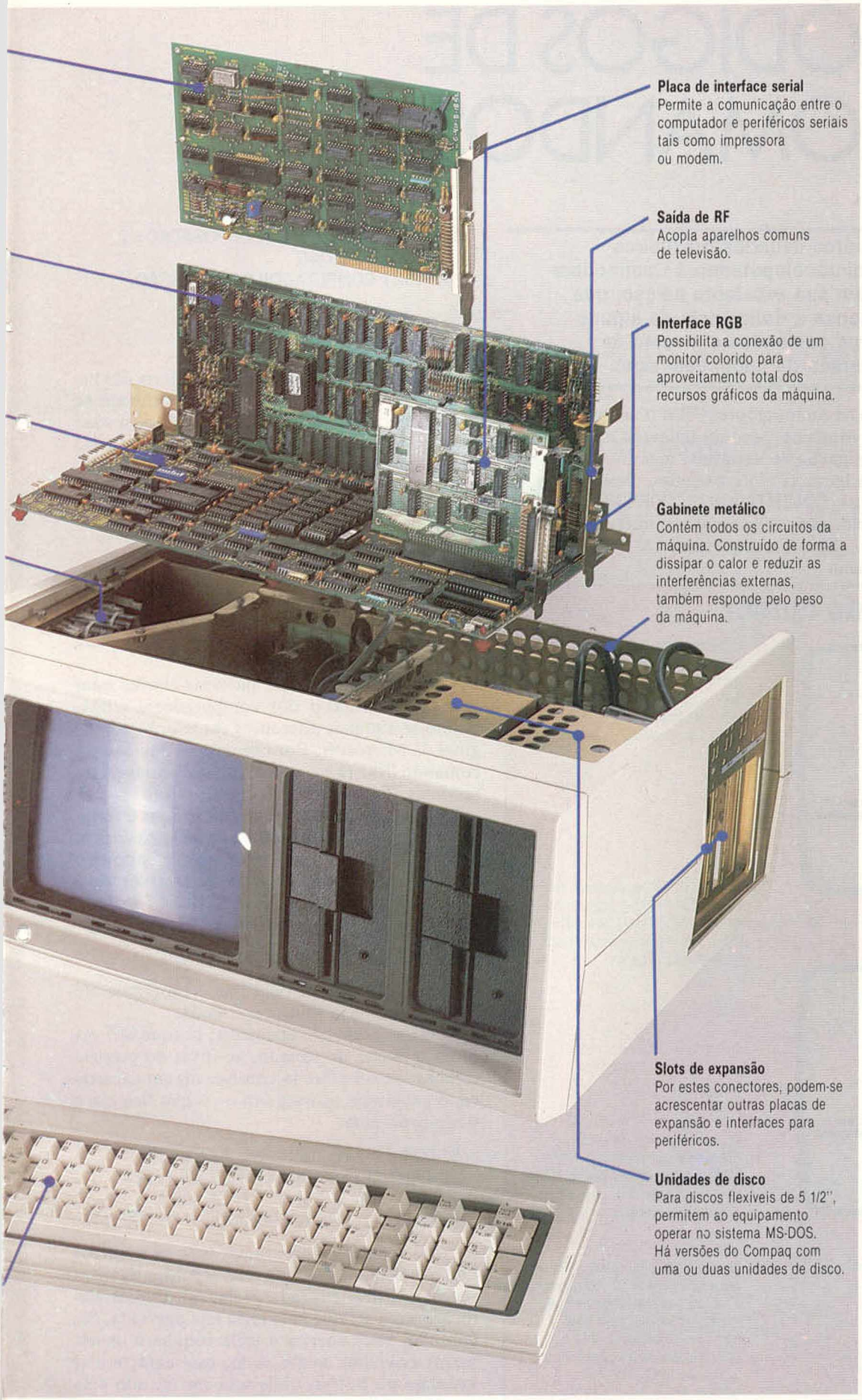
Placa da impressora
Com interface padrão Centronics, permite acoplamento de uma impressora paralela.

Placa de vídeo
Encaixa-se a um dos slots de expansão e fornece os circuitos necessários para acionar um monitor ou tevê comum.

Placa principal
Contém RAM de 256 Kbytes, o processador 8088 e os chips de entrada/saída. Há também slots disponíveis para chips extras, como o processador aritmético 8087.

Fonte de alimentação
Ao lado da entrada de energia encontra-se um ventilador, que refrigera o interior do equipamento.

Teclado padrão IBM
Idêntico ao do IBM PC, até na inclinação. À esquerda estão as teclas de função e à direita as de calculadora.



Placa de interface serial
Permite a comunicação entre o computador e periféricos seriais tais como impressora ou modem.

Saída de RF
Acopla aparelhos comuns de televisão.

Interface RGB
Possibilita a conexão de um monitor colorido para aproveitamento total dos recursos gráficos da máquina.

Gabinete metálico
Contém todos os circuitos da máquina. Construído de forma a dissipar o calor e reduzir as interferências externas, também responde pelo peso da máquina.

Slots de expansão
Por estes conectores, podem-se acrescentar outras placas de expansão e interfaces para periféricos.

Unidades de disco
Para discos flexíveis de 5 1/2", permitem ao equipamento operar no sistema MS-DOS. Há versões do Compaq com uma ou duas unidades de disco.

COMPAQ PLUS

MICROPROCESSADOR
Intel 8088.

CLOCK
4,7 MHz.

MEMÓRIA
256 Kbytes de RAM, expansível até 640 Kbytes, e 8 Kbytes de ROM.

SISTEMA OPERACIONAL
MS-DOS.

VÍDEO
Monitor de fósforo verde de 17,7 x 13,3 cm.
Modo texto, 25 linhas de 80 colunas; modo gráfico, 640 x 200 pixels.

TECLADO
Com inclinação ajustável, tem 81 teclas, sendo catorze de controle, dez de funções e dez de calculadora.

LINGUAGENS
BASIC (carregado a partir do disco-mestre) e todas as que rodam sob MS-DOS, como COBOL, FORTRAN, PASCAL.

INTERFACES
Saída paralela padrão Centronics, conector de RF e RGB. Há também slots de expansão para conexão de outras placas de interfaces.

DOCUMENTAÇÃO
Acompanham três manuais: guia de operação, guia de referência do MS-DOS e guia de referência de BASIC. Dos três, apenas o guia de operações mostra passo a passo todos os procedimentos. Para os iniciantes, talvez sejam necessários outros manuais auto-instrutivos.

CÓDIGOS DE COMANDOS

Os muitos usuários brasileiros dos microcomputadores Commodore elogiam sua excelente edição, que compensa a deficiência de alguns de seus comandos e mesmo de sua versão da linguagem BASIC.

Todos os computadores CBM (Commodore Business Machine) aceitam palavras longas como denominação de variáveis, mas só as duas primeiras letras são interpretadas. Desse modo, as palavras COMPUTADOR e CORPORACAO, por exemplo, são aceitas mas consideradas equivalentes. Em conseqüência, o resultado do programa

```
100 COMPUTADOR = 17 : CORPORACAO = 2
    * COMPUTACAO
200 PRINT COMPUTADOR,CORPORACAO
```

será

```
 34      34
```

Devido à forma como trabalha a maioria dos interpretadores BASIC, a velocidade se elevará se a inicialização do programa for feita com suas variáveis colocadas em ordem de importância. Consegue-se isso por meio da declaração de atribuições ou com o uso do comando DIM. Uma instrução como

```
10 DIM A$(10,24),K,L,TOTAL
```

não terá efeitos óbvios, mas é uma forma rápida de destacar as variáveis K, L e TOTAL. Faz também com que elas se tornem mais acessíveis ao interpretador; logo, incrementa a velocidade do programa.

Verificando a lista de palavras-chaves num manual do usuário dos computadores CBM, notam-se algumas diferenças em relação ao original da Microsoft. Por exemplo, a omissão do comando INKEY$ e o acréscimo das variáveis reservadas TIME$ e STATUS.

INKEY$, que tem a função de explorar o teclado, é substituído pelo comando GET. Assim como INKEY$, GET faz com que o primeiro caractere do buffer do teclado seja explorado e o transforma em seu valor ASCII. GET é usado mais freqüentemente em rotinas como

```
150 GET GT$ : IF GT$ = " "THEN 150
```

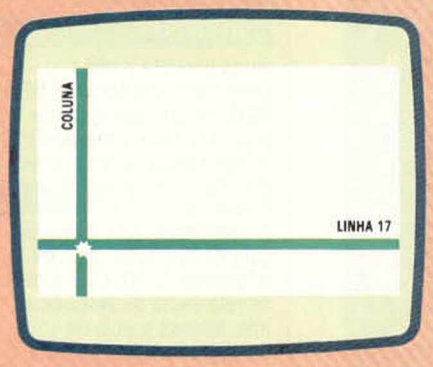
que tem a função de paralisar o programa até que alguma tecla seja pressionada.

Mas isso pode não acontecer, porque GET explora o buffer do teclado, ao invés do próprio teclado. Se o buffer já contiver algum caractere, o programa prosseguirá — o que fica claro neste programa:

```
50 FOR K = 1 TO 100 : PRINT K : NEXT K
60 PRINT "PRESSIONE QUALQUER TECLA"
150 GET GT$ : IF GT$ = " "THEN 150
200 PRINT "VOCE PRESSIONOU A TECLA";GT$
```

Se você rodar esse programa, verá os números de 1 a 100 aparecerem na tela, seguidos da mensagem de INPUT (entrada de dados). Depois, nada acontece até que uma tecla seja apertada. Se, contudo, você apertar a tecla enquanto os números estiverem aparecendo, esse caractere se instalará no buffer, onde será encontrado pelo comando GET.

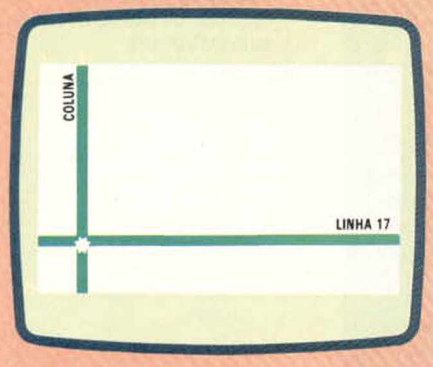
Posicionamento do cursor



A possibilidade de incluir os comandos de cursor numa matriz quantitativa facilita o desenho gráfico nos computadores Commodore, sobretudo quando acoplado aos poderosos comandos de edição.

```
100 PRINT AT(17,4)" * "
```

Se o BASIC da Commodore aceitasse o comando PRINT AT, o posicionamento do cursor seria simplificado. Mas essa limitação pode ser contornada pela programação do cursor.



Inicialize igualando a variável POSICAO$ a um HOME e 24 CRSR DOWNs (cursores para baixo). Então, coloque os parâmetros da linha e da coluna nesta expressão:

```
50 POSICAO$ = "SQQQQQQQQQQQQQQQQQQQQQQQQ"
100 PRINT LEFT$(POSICAO$,17)TAB(4-1)" * "
```

Se você tiver muitas formatações de tela a fazer, será aconselhável colocar o comando de posicionamento de cursor numa sub-rotina e então inicializar as variáveis LINHA e COLUNA com as posições de tela requeridas antes de chamar a sub-rotina.

```
50 POSICAO$ = "SQQQQQQQQQQQQQQQQQQQQQQQQ"
100 LINHA = 17: COLUNA = 4 : GOSUB 1000 : PRINT " * "
500 END
1000 PRINT LEFT$(POSICAO$,LINHA)TAB (COLUNA-1); : RETURN
```

No caso, não haverá pausa na execução do programa, o que incomoda o usuário e até impossibilita jogos em que certas teclas são mais freqüentemente acionadas. A solução está em anular ([Reset]) o buffer antes de o comando GET ser executado. Para isso, basta inserir no programa

```
149 POKE KBPTR,0
```

onde KBPTR é o endereço do contador seqüencial do teclado (198 no Commodore 64).

A variável reservada TIME$, cuja abreviatura é TI$, é o clock do sistema; quando o computador for ligado ele estará em 000000 e contará o tempo em horas, minutos e segundos até 235959 — ou seja, 23 horas, 59 minutos e 59 segundos, desde que o computador foi ligado. Então ele se anulará, voltando para o começo.

Associado a TI$ há TI, uma subdivisão numérica de TI$. TI conta o tempo em 60 avos de segundo (chamados "jiffys"). Os jiffys variam entre 0 e 5183999 (60 * 60 * 60 * 24-1). TI depende inteiramente de TI$ e não pode ser inicializado por si. Apenas TI$ se inicializa.

STATUS, abreviado por ST, é uma variável definida pelo sistema. Quando se detecta um erro numa operação de entrada ou saída de dados, o valor de ST corresponde ao número que indica o tipo de erro encontrado. Normalmente, ele será escrito como

```
330 IF ST > 0 THEN GOSUB 30000 : REM E/S ERRO
    NO TRATAMENTO DE ARQUIVOS
:
30000 REM MENSAGENS DE ERRO
30100 IF ST = 16 PRINT "ERRO IRRECUPERAVEL
      DE LEITURA"
```

CMD é uma instrução muito útil dos computadores Commodore. Sua função está em direcionar o conteúdo da tela para qualquer outro canal de saída. Ela tem muitas aplicações, como

```
OPEN 4,4 : CMD 4 : LIST
```

que lista o programa na impressora, ao invés de na tela. Quando a listagem estiver completa, deverá ser executado:

```
PRINT # 4 : CLOSE 4
```

Utiliza-se CMD num programa sempre que se quer imprimir o que estiver na tela. Suponha que GOSUB 3000, em seu programa, faça com que uma mensagem ou alguns dados sejam escritos na tela. Para comandar a cópia dessa tela para a impressora, faça o seguinte:

```
OPEN 4,4 : CMD 4 : GOSUB 3000 : PRINT #
  4 : CLOSE 4
```

Embora um ponto de interrogação (?) seja aceito como abreviação do comando PRINT, não se pode abreviar o comando PRINT# por ?# e sim por pR.

Na prática, qualquer comando dos computadores Commodore pode ser abreviado, usando-se para isso sua primeira letra em minúscula seguida pela segunda em maiúscula. Quando dois ou mais comandos têm as duas primeiras letras iguais, a abreviação deve ser feita com as duas primeiras letras em minúsculas, seguidas da terceira, em maiúscula. Por exemplo, READ, RESTORE e RETURN abreviam-se rE, reS e reT.

A edição, reforçada pelo sistema operacional facilitado ao usuário, é a mais significativa característica dos computadores Commodore. Vem sendo usada desde o lançamento do PET e é até hoje considerada a melhor de todas.

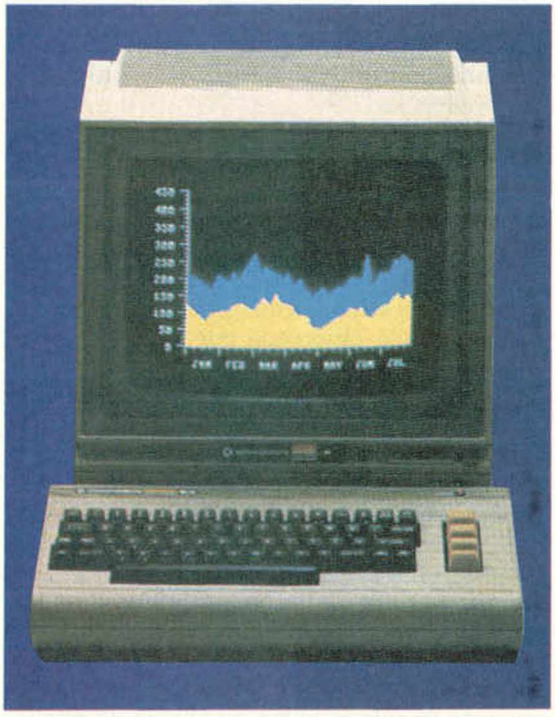
Para editar uma linha do programa, basta listá-la, movimentar o cursor para qualquer ponto da linha, editar o texto e pressionar a tecla [Return]. Independentemente da localização do texto na tela ou do cursor na linha, quando [Return] é pressionada, o texto contido na linha em que se localiza o cursor entra no sistema como se tivesse acabado de ser digitado.

Uma vantagem dessa edição está na facilidade de reprodução. Suponha a necessidade de digitar estas duas linhas:

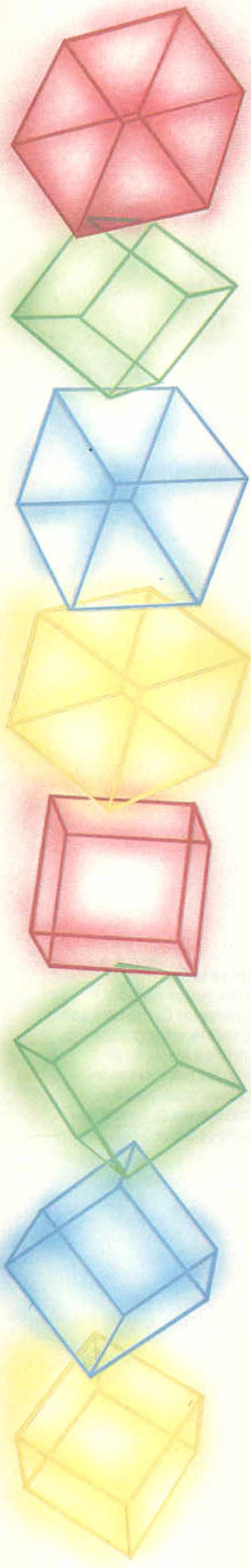
```
100 IF INT(NUMERO/INDICE—TAXA) = 5 THEN
    3000
200 IF INT(NUMERO/INDICE—TAXA) = 7 THEN
    3800
```

Digite a primeira linha e acione a tecla [Return]; então, mantendo esse texto na tela, mova o cursor para cima, modifique o número da linha 100 para 200, troque o 5 por 7, troque 3000 por 3800 e pressione de novo [Return]. A linha 100 estará guardada na memória e as transformações de edição farão a linha 200 surgir na tela.

Esse processo se repete quantas vezes necessário. Trata-se apenas de um dos truques permitidos pelos recursos de edição dos computadores Commodore, mas já demonstra como seu uso é fácil e direto.



Telas versáteis
Um dos pontos altos dos equipamentos Commodore está nos seus recursos gráficos, que permitem a execução de avançados softwares para empresas, como planilhas, processadores de texto e bancos de dados.



A REVOLUÇÃO CUBISTA

Os princípios geométricos usados no programa para traçar figuras tridimensionais (projeções em perspectiva) fornecem uma base para que se desenvolvam alguns gráficos animados simples.

O programa aqui apresentado utiliza princípios geométricos básicos para criar projeções em perspectiva de objetos que podem ser observados de qualquer ângulo ou distância.

Todos os dados dos objetos são armazenados no programa por meio de instruções DATA. Essas instruções contêm as coordenadas tridimensionais de cada vértice do objeto. Para cada ponto há também um número que esclarece se ele está no início de uma linha ou em seu final (o que indica se a linha será traçada a partir do ponto ou até ele).

Converter essas coordenadas tridimensionais em valores bidimensionais representando pontos na tela é questão de simples porém extenso trabalho matemático. As coordenadas x e y de cada ponto (no plano da tela do vídeo) são divididas por um fator que representa a distância do objeto em relação ao observador. Além disso, aplica-se à coordenada resultante um fator de escala apropriado ao sistema de coordenadas do micro. A modificação do fator de distância faz o objeto diminuir ou aumentar de tamanho, conforme se aproxime ou se afaste do observador.

Usa-se, ainda, uma terceira constante para modificar o efeito da projeção em perspectiva. Aumentando o valor dessa constante, a imagem em perspectiva do objeto é exagerada, como se ele estivesse sendo observado por meio de lente grande-angular. Diminuindo a constante, obtém-se o efeito de achatamento da imagem, como se estivesse sendo observada por meio de teleobjetiva.

Além de criar uma imagem em perspectiva do objeto, o programa possibilita girá-la de modo que possa ser observada de qualquer ângulo. Isso se realiza por meio de trigonometria simples. Os eixos sofrem uma rotação até o ângulo desejado, de modo que, quando se faz a projeção em perspectiva, a imagem na tela parece ter girado. Consegue-se esse efeito com uma rotação em torno do eixo y ou do eixo x. No primeiro caso, o ponto de observação parece girar na horizontal em torno do objeto; no segundo, o ponto de observação se coloca acima ou abaixo do objeto.

Embora crie todos esses efeitos, nosso programa é extremamente simples. O controle do ponto de observação se faz pelas teclas numéricas de 1 a 8. Essas teclas darão, respectivamente: um movimento do ponto de observação para a esquerda (tecla [1]), para a direita (tecla [2]), para cima (tecla [3]), para baixo (tecla [4]), em direção ao objeto (tecla [5]) ou afastando-se dele (tecla [6]), um aumento (tecla [7]) no efeito de perspectiva (olho-de-peixe) ou uma diminuição (tecla [8]) no efeito de perspectiva (teleobjetiva).

As coordenadas tridimensionais dos pontos (vértices) são armazenadas numa única matriz: V(6,50).

A alteração dessa matriz e a alteração das constantes usadas na transformação da perspectiva são feitas por meio de uma série de sub-rotinas. Cada vez que se pressiona uma tecla, a imagem na tela é apagada por um comando HGR2 (no Apple) ou CLS (no TK 90X); a alteração nas coordenadas dos vértices se realiza por uma sub-rotina apropriada; e a nova imagem é, então, desenhada.

A transformação da perspectiva ocorre na sub-rotina que transforma a imagem. Essa sub-rotina percorre cada conjunto de coordenadas tridimensionais, transforma-as em coordenadas bidimensionais e as traça na tela (movendo-se até os vértices ou desenhando linhas, de acordo com um parâmetro adicional introduzido junto com cada coordenada tridimensional).

Os dados são armazenados em instruções DATA no final do programa, com quatro valores para cada vértice do objeto. O número total de vértices é estabelecido na primeira linha do programa (atributo da variável N). Isso deve ser modificado para adaptar-se a seus dados. Os dados que fornecemos criam um cubo com uma linha diagonal atravessando uma das faces.

Cada conjunto de quatro valores está nesta ordem: parâmetro para desenhar ponto ou traçar linha, coordenada x, coordenada y, coordenada z. Os valores são calculados traçando-se mentalmente o contorno do objeto em três dimensões. Com o auxílio de uma caneta imaginária, vá até cada vértice do contorno do objeto. Se sua caneta for movida até um vértice sem desenhar uma linha, o primeiro valor será 4. Se o primeiro valor for 5, será desenhada uma linha do vértice precedente até esse vértice.

A origem das coordenadas (0,0) está no centro da tela. É melhor fazer dela o centro do objeto. O eixo x é o eixo horizontal, com os valores positivos aumentando para a direita. O eixo y é o vertical, com os valores positivos aumentan-

do de baixo para cima. O eixo z é o que vai de fora para dentro da tela. O sentido positivo desse eixo também vai para a tela.

Procure usar os menores valores possíveis para as coordenadas X, Y e Z dos vértices de seu objeto. O ajuste inicial do efeito de perspectiva e de distância do ponto de observação deve levar em conta que uma largura de objeto por volta de 10 irá preencher toda a tela. Isso pode ser modificado pela alteração do fator de escala usado na transformação da perspectiva.

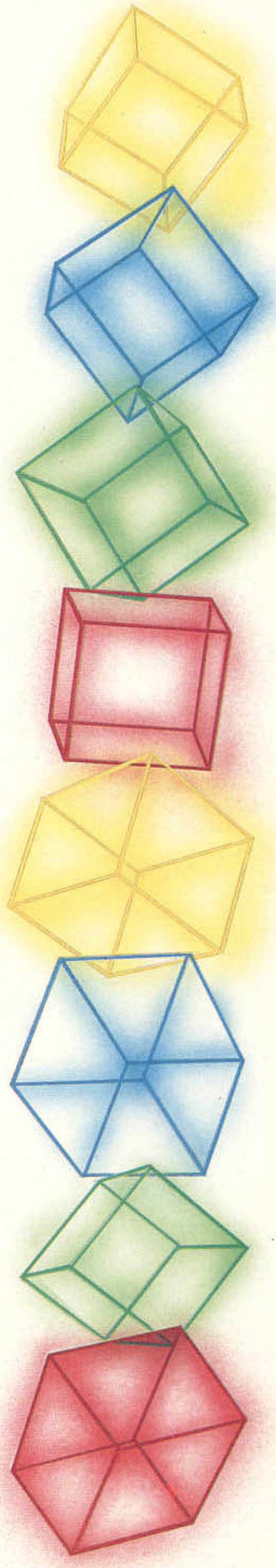
Faça primeiramente experiências com as formas mais simples. Procure usar poucos pontos. Quando tiver dominado a construção de objetos sólidos simples (pirâmides, cubos etc.), você passará para outros, mais complexos.

Versão Apple e TK 2000

```
10 N = 16: DIM V(6,50):D = 10:P =
    0.5:SI =  SIN (0.09):CO =  COS
    (0.09): HGR2
20  FOR I = 1 TO N: READ V(1,I),V
    (2,I),V(3,I),V(4,I): NEXT I
30  HCOLOR= 3: GOSUB 300
40  GET I$: HGR2
50  ON  VAL (I$) GOSUB 1000,2000,
    3000,4000,5000,6000,7000,800
    0
60  GOTO 30
300  FOR I = 1 TO N:V(5,I) = V(2,
    I) * 300 / (P * V(4,I) + D):
    V(6,I) = V(3,I) * 300 / (P *
    V(4,I) + D): NEXT I
310  FOR I = 1 TO N
320  IF V(1,I) = 4 THEN  HPLOT (V
    (5,I) + 140),(V(6,I) + 95)
330  IF V(1,I) = 5 THEN  HPLOT (V
    (5,I - 1) + 140),(V(6,I - 1)
     + 95) TO (V(5,I) + 140),(V(
    6,I) + 95)
340  NEXT I: RETURN
1000  FOR I = 1 TO N:X = V(2,I) *
    CO - V(4,I) * SI:Z = V(4,I) *
    CO + V(2,I) * SI:V(2,I) = X:
    V(4,I) = Z: NEXT I: RETURN
1010  PRINT  CHR$ (9);"40N"
1020  LIST 10,9999
1030  PRINT  CHR$ (9);"PR#0"
2000  FOR I = 1 TO N:X = V(2,I) *
    CO + V(4,I) * SI:Z = V(4,I) *
    CO - V(2,I) * SI:V(2,I) = X:
    V(4,I) = Z: NEXT I: RETURN
3000  FOR I = 1 TO N:Y = V(3,I) *
    CO + V(4,I) * SI:Z = V(4,I) *
    CO - V(3,I) * SI:V(3,I) = Y:
    V(4,I) = Z: NEXT I: RETURN
4000  FOR I = 1 TO N:Y = V(3,I) *
    CO - V(4,I) * SI:Z = V(4,I) *
    CO + V(3,I) * SI:V(3,I) = Y:
    V(4,I) = Z: NEXT I: RETURN
5000 D = D * 0.9: RETURN
6000 D = D / 0.9: RETURN
7000 P = P / 0.9: RETURN
8000 P = P * 0.9: RETURN
9000  DATA  4,1,1,1,5,1,1,-1,5,-1
    ,1,-1,5,-1,1,1,5,1,1,1
9010  DATA  5,1,-1,1,5,1,-1,-1,5,
    -1,-1,-1,5,-1,-1,1,5,1,-1,1
9020  DATA  4,1,-1,-1,5,1,1,-1,5,
    -1,-1,-1,5,-1,1,-1
9030  DATA  4,-1,1,1,5,-1,-1,1
```

Versão TK 90X

```
  10 LET N=16: DIM V(6,50)
  20 LET D=10: LET P=0.5
  30 LET SI=SIN 0.09: LET CO=COS
0.09
  40 FOR I=1 TO N
  50 READ V(1,I),V(2,I),V(3,I),V
(4,I)
  60 NEXT I
  70 INVERSE 0: GOSUB 300
  80 IF INKEY$<>"" THEN GOTO 80
  90 IF INKEY$="" THEN GOTO 90
 100 LET I$=INKEY$
 110 CLS
 120 GOSUB VAL I$*1000
 130 GOTO 70
 300 FOR I=1 TO N: LET V(5,I)=V(
2,I)*300/(P*V(4,I)+D): LET V(6,I
)=V(3,I)*300/(P*V(4,I)+D): NEXT
I
 310 FOR I=1 TO N
 320 IF V(1,I)=4 THEN PLOT V(5,I
)+127,V(6,I)+87
 330 IF V(1,I)=5 THEN DRAW V(5,I
)-V(5,I-1),V(6,I)-V(6,I-1)
 340 NEXT I: RETURN
1000 FOR I=1 TO N: LET X=V(2,I)*
CO-V(4,I)*SI: LET Z=V(4,I)*CO-V(
2,I)*SI: LET V(2,I)=X: LET V(4,I
)=Z: NEXT I: RETURN
2000 FOR I=1 TO N: LET X=V(2,I)*
CO+V(4,I)*SI: LET Z=V(4,I)*CO-V(
2,I)*SI: LET V(2,I)=X: LET V(4,I
)=Z: NEXT I: RETURN
3000 FOR I=1 TO N: LET Y=V(3,I)*
CO+V(4,I)*SI: LET Z=V(4,I)*CO-V(
3,I)*SI: LET V(3,I)=Y: LET V(4,I
)=Z: NEXT I: RETURN
4000 FOR I=1 TO N: LET Y=V(3,I)*
CO-V(4,I)*SI: LET Z=V(4,I)*CO+V(
3,I)*SI: LET V(3,I)=Y: LET V(4,I
)=Z: NEXT I: RETURN
5000 LET D=D*0.9: RETURN
6000 LET D=D/0.9: RETURN
7000 LET P=P/0.9: RETURN
8000 LET P=P*0.9: RETURN
9000 DATA 4,1,1,1,5,1,1,-1,5,-1,
1,-1,5,-1,1,1,5,1,1,1
9010 DATA 5,1,-1,1,5,1,-1,-1,5,-
1,-1,-1,5,-1,-1,1,5,1,-1,1
9020 DATA 4,1,-1,-1,5,1,1,-1,5,-
1,-1,-1,5,-1,1,-1
9030 DATA 4,-1,1,1,5,-1,-1,1
```



MÉTODOS ALTERNATIVOS

Embora os três tipos básicos de porta (E, OU e NÃO) resolvam qualquer problema lógico, as portas lógicas $\overline{E}$ e $\overline{OU}$ oferecem métodos alternativos que simplificam o desenho de circuitos.

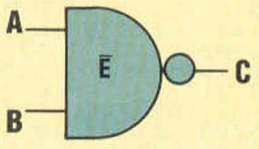
Se podemos resolver todos os problemas lógicos com o uso de portas E, OU e NÃO, por que aprender outros tipos de porta? O motivo é a redução do custo de fabricação do circuito que essas portas permitem, quer isoladamente, quer combinadas com outras. Isso porque elas simplificam o sistema de conexão ou oferecem soluções mais refinadas. Todos os problemas lógicos podem ser resolvidos pelo uso de uma das seguintes técnicas:

a) portas E, OU e NÃO juntas;
b) somente portas $\overline{E}$;
c) somente portas $\overline{OU}$;
d) uma combinação de todas.

A função de cada porta é melhor descrita por sua tabela de validação:

A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

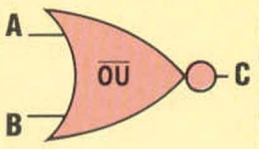
PORTA $\overline{E}$



$\overline{E}$ é a representação de NÃO E, e a comparação dessa tabela de validação com a tabela usada para a porta E permite notar que na coluna da saída todos os 1 foram substituídos por 0, e vice-versa.

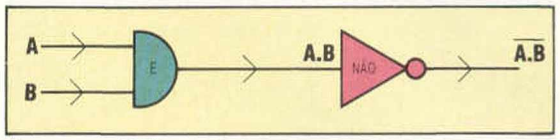
A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

PORTA $\overline{OU}$

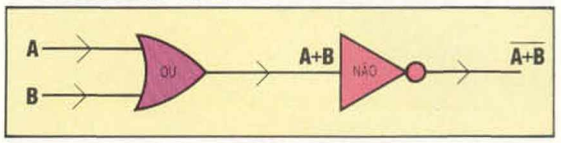


Do mesmo modo, $\overline{OU}$ é a representação de NÃO OU, e a comparação das colunas de saída (coluna C) para essa tabela e para a tabela da porta OU mostra que todos os 1 e 0 foram negados.

Não há símbolos especiais para as operações $\overline{E}$ e $\overline{OU}$ na álgebra booleana, mas é possível representar cada função booleana com os símbolos E, OU e NÃO, examinados anteriormente. Uma porta $\overline{E}$ equivale a este circuito simples:



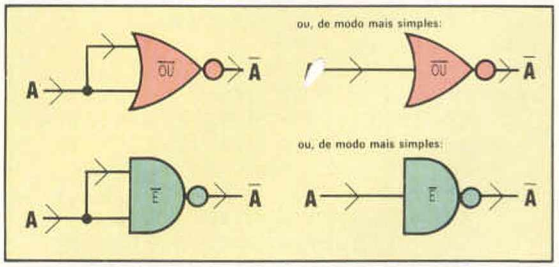
E a porta $\overline{OU}$ equivale a uma porta OU seguida de uma porta NÃO:



O uso de $\overline{E}$ e $\overline{OU}$

Assim como podemos desenhar circuitos E/OU/NÃO que equivalem a $\overline{E}$ e $\overline{OU}$, também é possível representarmos cada uma dessas três portas básicas em termos de uma série de portas $\overline{OU}$ ou de uma série de portas $\overline{E}$.

Portas NÃO: A negação pode ser obtida conectando-se ambas as entradas, com o uso de uma porta $\overline{OU}$ ou de uma porta $\overline{E}$:

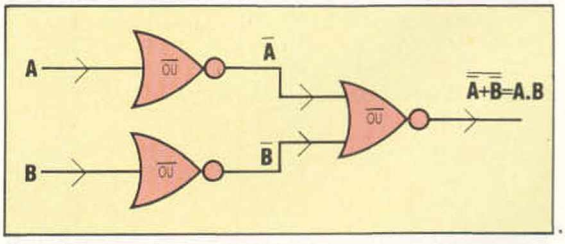


Portas E: Em termos de álgebra booleana, a saída de uma porta E com entradas A e B será A . B. No entanto, essa expressão pode ser convertida numa forma mais útil:

$$A \cdot B = \overline{\overline{A} \cdot \overline{B}} \qquad \text{(pois } A = \overline{\overline{A}}\text{)}$$

$$= \overline{\overline{A} + \overline{B}} \qquad \text{(lei de De Morgan)}$$

Assim, desenhamos o circuito colocando NÃO(A) e NÃO(B) diante de uma porta $\overline{OU}$:

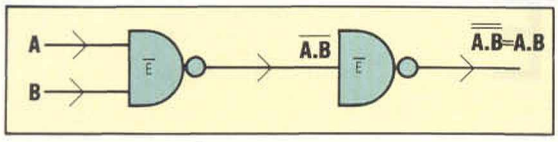


Também podemos criar uma porta E usando portas $\overline{E}$. A saída de uma porta $\overline{E}$ é $\overline{A \cdot B}$. Se es-

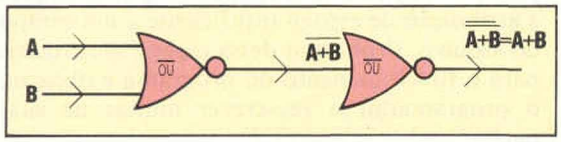
sa saída for negada, obteremos

$$\overline{\overline{A.B}} = A.B$$

Assim, o circuito será:



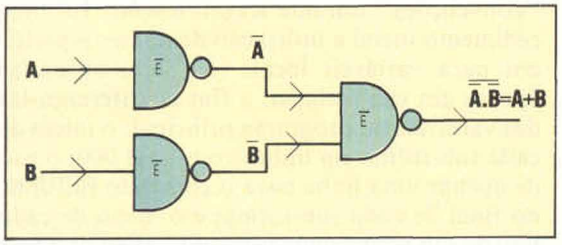
Portas OU: Assim como o encadeamento de duas portas Ē equivale a uma porta E, do encadeamento de duas portas $\overline{OU}$ resultará um circuito equivalente a uma porta OU:



A saída para uma porta OU é A + B. Usando as regras da álgebra booleana, modificamos essa saída para uma forma Ē:

$$A + B = \overline{\overline{A}} + \overline{\overline{B}}$$
$$= \overline{\overline{A}.\overline{B}}$$

Assim, o circuito correspondente que usa as portas Ē será:

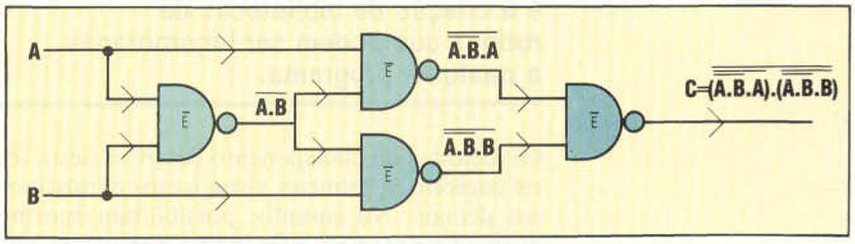


Para obter um circuito que utiliza apenas elementos Ē ou $\overline{OU}$, podem-se ainda seguir os métodos de simplificação já vistos, mas antes é preciso modificar a expressão booleana final a uma forma apropriada. Para circuitos que incorporam as portas Ē, utilizam-se as regras da álgebra booleana a fim de criar uma expressão que consista em grupos de Es ligados por meio de OUs; e usa-se o teorema de De Morgan seguidas vezes até que a expressão esteja completamente na forma Ē. Para circuitos da forma $\overline{OU}$, empregam-se regras semelhantes. Em demonstrar como essas regras são usadas, convém examinar a porta OU exclusivo (XOU). A saída dessa porta pode ser definida pela expressão $C = \overline{A.B}.(A + B)$.

É possível converter essa expressão de modo a constituir o circuito para a porta XOU apenas por meio de portas Ē. Em primeiro lugar, obtêm-se grupos de Es conectados por meio de OUs:

$$C = \overline{A.B}.(A + B)$$
$$= (\overline{A.B}.A) + (\overline{A.B}.B) \quad \text{(multiplicação dos parênteses)}$$
$$= \overline{\overline{\overline{A.B}.A}}.\overline{\overline{\overline{A.B}.B}} \quad \text{(teorema de De Morgan)}$$

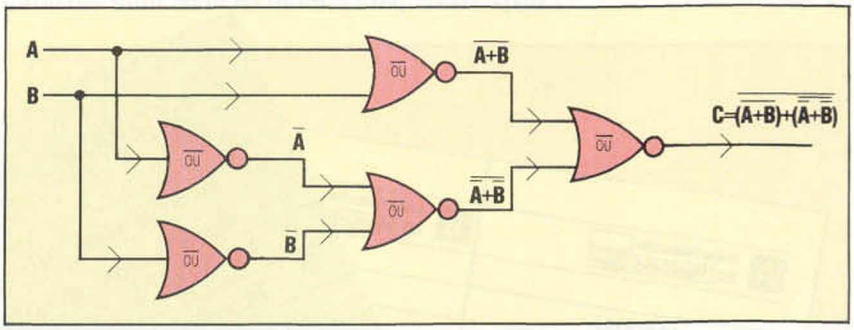
Ao desenhar o circuito para uma expressão complexa como essa, é melhor iniciar a partir da saída e trabalhar voltando em direção às entradas. Tente seguir com essa orientação o diagrama de circuito abaixo para ver como ele foi construído.



Para a forma OU, é necessário mais uma vez começar com a expressão simplificada original para a porta XOU e modificá-la para formar grupos de OUs conectados por meio de Es. Esse primeiro passo pode ser feito com o teorema de De Morgan na parte da esquerda:

$$C = \overline{A.B}.(A + B)$$
$$= (\overline{A} + \overline{B}).(A + B)$$
$$= \overline{\overline{(\overline{A} + \overline{B})} + \overline{(A + B)}}$$

Também nesse caso a conversão da expressão num diagrama de circuitos é mais fácil se realizada a partir da saída em direção às entradas.



Respostas do exercício anterior

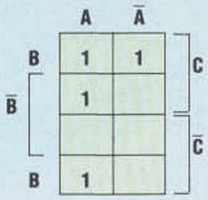
1a)

Entradas			Saída
A	B	C	P
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

b) A expressão booleana para P é:

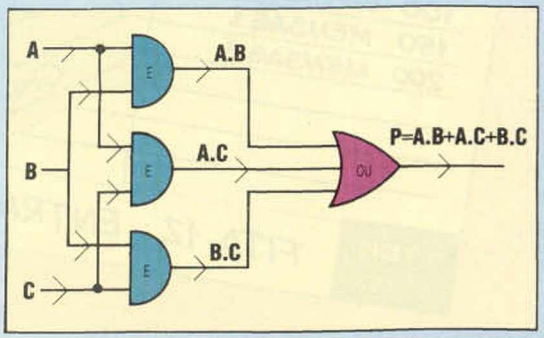
$$P = \overline{A}.B.C + A.\overline{B}.C + A.B.\overline{C} + A.B.C$$

Podemos simplificar com o uso de um mapa-k:



Assim, $P = A.B + A.C + B.C$

c) O circuito é:



BIBLIOTECA CIRCULANTE

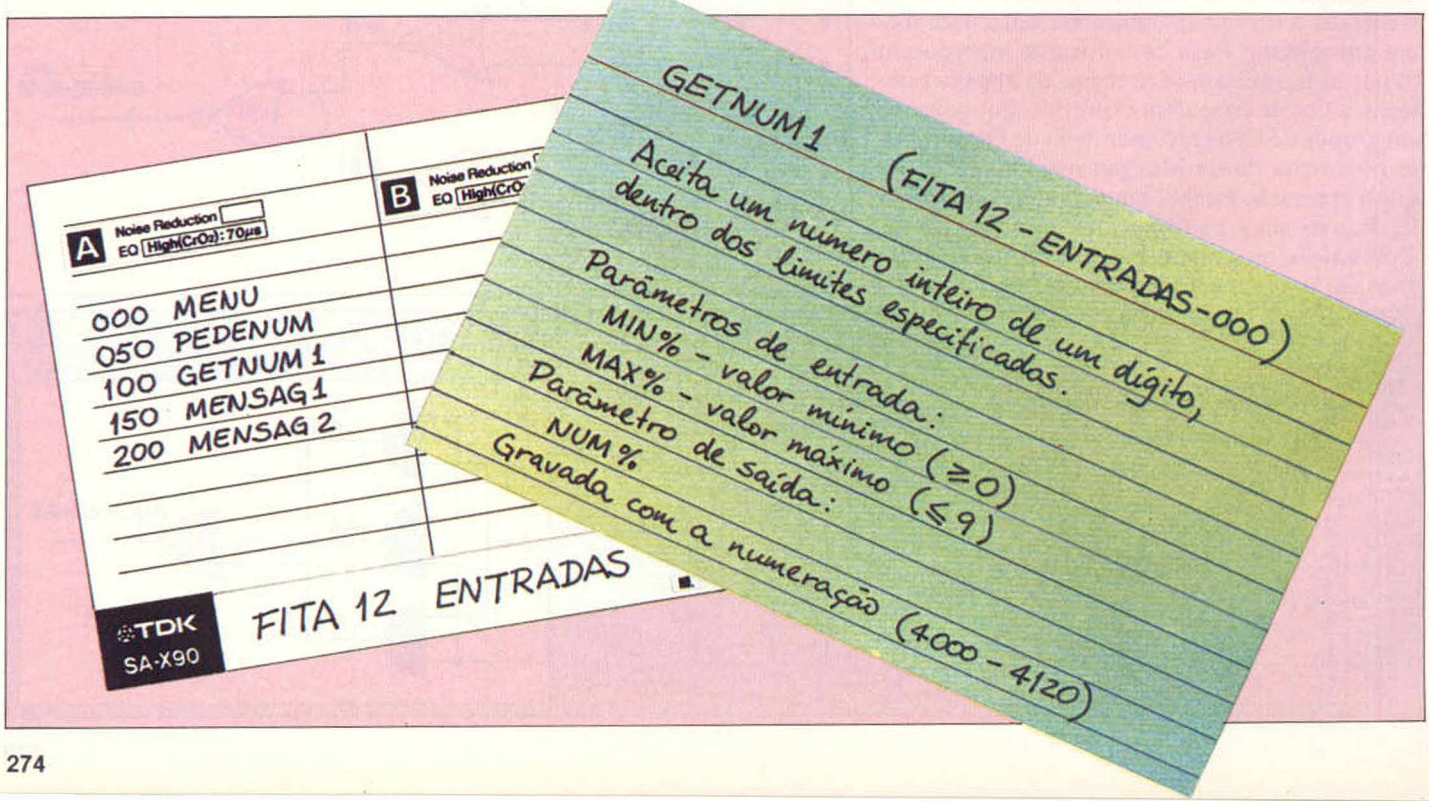
De enorme utilidade são os métodos para economizar tempo e esforço em programação. Uma dessas técnicas é a criação de bibliotecas de rotinas que podem ser incorporadas a qualquer programa.

Os métodos de planejamento descritos nesta série parecem, à primeira vista, lentos e trabalhosos demais. No entanto, possibilitam enorme economia de tempo, não apenas durante a codificação, mas também na depuração dos erros. Em geral, os programas criados no teclado, sem nenhum planejamento, possuem estruturas e algoritmos desnecessariamente complicados. Isso significa que exigem muito tempo para ser desenvolvidos, estão mais sujeitos a erros e, por fim, acarretam mais trabalho quando testados e corrigidos, em razão de sua lógica defeituosa. O planejamento prévio de um programa simplifica bastante sua estrutura e seus algoritmos, diminuindo os erros de codificação e facilitando os testes e a depuração.

Mais importante ainda, o planejamento é indispensável para a montagem de uma estrutura de arquivo ou de controle que não se revele depois inteiramente inadequada — por exemplo, a atribuição de espaço insuficiente a um campo do arquivo. Problemas dessa espécie são cruciais para o funcionamento do programa e obrigam o programador a reescrever muitas de suas partes.

Aperfeiçoando a datilografia, os usuários que possuem micros com teclado profissional, semelhante ao das máquinas de escrever, são capazes de aumentar a velocidade dos toques. Fora isso, não existe nenhum outro método para tornar mais rápida a digitação das linhas do programa. Mas é possível acelerar bastante o processo de codificação dos programas. O recurso mais simples é o de inventar, adotar e usar inúmeras "convenções" durante a codificação. Tal procedimento inclui a utilização de nomes específicos para variáveis locais (ou seja, utilizadas apenas em sub-rotinas), a fim de diferençá-las das variáveis do programa principal; o início de cada sub-rotina em linhas com final 000; o uso de apenas uma linha para o comando RETURN, no final de cada sub-rotina; e o início de cada tipo de sub-rotina num grupo específico de linhas (manipulação de arquivos entre 9000 e 9999, utilitários a partir de 50000 etc.).

Sistema uniforme
De nada vale montar uma biblioteca de sub-rotinas sem um sistema de documentação uniforme. Isso se aplica com mais força aos usuários de fita cassete — é muito irritante ter de descobrir o conteúdo de fitas sem indicação alguma, carregando-as no micro.



O emprego dessas convenções apresenta muitas vantagens. Não é necessário procurar laboriosamente determinada rotina na listagem, pois cada tipo está sempre no mesmo lugar; não é preciso preocupar-se com a atribuição do mesmo nome para uma variável no programa principal e numa sub-rotina, porque o nome indicará se a variável é local ou global.

Biblioteca de programas

Essas técnicas de codificação também facilitam bastante a organização de uma biblioteca de programas. Uma bem organizada biblioteca de sub-rotinas permite reduzir pela metade o tempo de codificação de um programa extenso.

A melhor maneira de começar uma biblioteca é examinar todos os programas disponíveis e, então, selecionar as sub-rotinas bem escritas e de emprego geral (rotinas de entrada e saída, de conversão de maiúsculas em minúsculas etc.). Cada rotina deve ser gravada como um arquivo separado. Estes serão agrupados conforme a função (se forem armazenados em fita, cada grupo de funções é gravado em fita separada), recebendo nomes característicos para melhor identificação. Mantenha um índice ou um banco de dados com os nomes dos arquivos e a descrição de cada rotina.

Tenha certeza de que todas as rotinas da biblioteca foram testadas e depuradas. Certifique-se de que rejeitarão quaisquer valores de entrada inadequados, pois serão usadas em programas para os quais não foram especialmente criadas. Verifique também se os valores de saída causarão problemas no programa que os utilizar. Faça com que cada rotina seja o mais eficiente possível e inclua algumas linhas com comentários para facilitar sua compreensão, posteriormente. Aumente sua coleção apenas o necessário e não se esqueça de numerar as linhas das rotinas de sua biblioteca conforme as convenções estabelecidas (evitando assim renumerá-las quando forem usadas num novo programa).

Para formar o programa completo é preciso um meio de agrupar várias rotinas. Quando se usa linguagem compilada, os vários módulos são unidos com auxílio de um programa tipo link-loader, tornando-os um único programa executável. Os programadores que não dispõem de um compilador de linguagem devem utilizar-se dos comandos MERGE e RENUM, já incorporados a algumas versões do BASIC ou disponíveis no mercado como utilitários.

Quando quiser acrescentar uma rotina ao novo programa, você precisa carregar o programa, decidir onde irá introduzir a rotina e certificar-se de que existe para isso um número suficiente de linhas não utilizadas. Se necessário, renumere a rotina, para que ela ocupe o espaço que lhe cabe. Use então o comando MERGE para unir os dois programas. Confira se tudo está funcionando direito e grave o novo programa com a rotina de biblioteca já incorporada.

MERGE

- O TK 90X possui o comando MERGE incorporado a seu BASIC. Com o programa a ser completado já na memória do micro, comanda-se MERGE "nome da rotina a ser carregada" e introduz-se o programa complementar do gravador. Caso haja sobreposição de linhas, aquela que está sendo carregada prevalece sobre a outra, que é anulada.
- O Apple, o TK 85, o CP 2000 e alguns outros micros, que não possuem um comando MERGE residente, dispõem de utilitários com rotinas MERGE, RENUM e outras, em discos ou fitas. O RENUM também está disponível na forma de utilitário para o TK 90X.

Trabalho em grupo

É comum que um grupo de programadores trabalhe em conjunto num projeto — seja numa empresa, seja numa escola ou num clube de micro. A maior parte do que foi dito sobre planejamento de programas e sobre a eficiência do programador é de especial importância para o trabalho em grupo. Muitos desses conceitos e idéias de programação estruturada foram desenvolvidos para que se dividisse a tarefa de produção de programas. Isso permite aos programadores trabalharem simultaneamente em diferentes partes do mesmo programa, terminando-o com mais rapidez.

Para que programadores em BASIC possam trabalhar em grupo, é essencial que estejam de acordo quanto às convenções que serão utilizadas na codificação. Uma vez estabelecido o plano geral do programa, o programador de cada módulo precisa saber o seguinte:

1) Em que consistem os arquivos e como serão organizados.
2) Quais as convenções utilizadas para dar nome às variáveis. As variáveis mais importantes — como as matrizes, que serão utilizadas no programa todo — têm seus nomes previamente definidos. É preciso estabelecer também uma convenção para variáveis locais e denominar com antecedência as variáveis que passam de um módulo para outro, ou achar um meio de garantir que cada uma delas seja única. Para isso, adiciona-se, por exemplo, o número do módulo em que ela é definida ao seu final.
3) Quais as rotinas disponíveis para o grupo, a função e o formato de cada uma, o nome de suas variáveis e o modo pelo qual foram testadas ou depuradas.
4) De que modo estão organizadas as rotinas de controle de erros; se cada rotina corrige seus próprios erros ou apenas os indica para que sejam corrigidos pela rotina de controle do programa mais abrangente.
5) A função exata de cada módulo.
6) O tipo de dado e a faixa de valores que cada módulo individual deve aceitar como entrada e devolver como saída.

ELETRÔNICA POPULAR

Adaptando micros ingleses versáteis e de baixo custo, a Microdigital firmou-se na produção de equipamentos de consumo basicamente popular e garantiu a liderança do mercado nacional.

Com mais de 100.000 unidades vendidas no Brasil desde sua fundação, em 1981, a Microdigital Eletrônica é o maior fabricante nacional de microcomputadores, responsável por 60% do mercado. Investindo 5% de seu faturamento anual (10 milhões de dólares em 1984) em pesquisa e desenvolvimento, a empresa contava, no início de 1985, com quatrocentos funcionários, três unidades industriais (duas em São Paulo e uma em Manaus), seis unidades de serviços técnicos e mais de setecentos revendedores. No mercado internacional, suas vendas atingiram 1,5 milhão de dólares em 1984, a maior parte para a Argentina, onde estabeleceu mais de trezentos pontos de venda e comercializou 10.000 aparelhos.

Os produtos desenvolvidos pela Microdigital têm grande aceitação entre estudantes e profissionais liberais.

A Microdigital inaugurou sua produção com o modelo TK 82, seguido pouco depois por uma versão com design reformulado, o TK 82C, ainda em 1981. Desenvolvido a partir do micro ZX 80, da Sinclair, esse equipamento foi substituído, no final de 1982, pelo TK 83. Com 8 Kbytes de ROM e programável em linguagem BASIC ou ASSEMBLY, o TK 83 vem equipado com teclado de membrana e duas interfaces para impressora e monitor de vídeo. De custo menor, esse micro teve larga penetração junto ao público iniciante na informática.

Dispondo de mais recursos que os modelos anteriores, o TK 85 chegou ao mercado em 1983, com ROM de 10 Kbytes. Seu teclado emborrachado, tipo chiclete, compõe-se de quarenta teclas no padrão QWERTY e permite acesso a 160 funções.

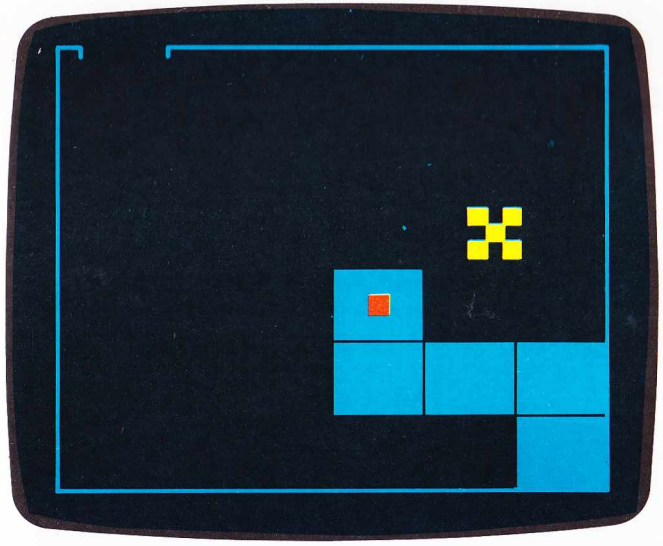
Embora deixasse de fabricar o TK 83 a partir de 1985, a empresa continuou lançando periféricos compatíveis com esse modelo e com o TK 85, entre os quais o gerador de som programável (PSG, Programmable Sound Generator) — capaz de criar inúmeros efeitos sonoros, de notas musicais a explosões —, além de interfaces para impressoras que utilizam o padrão Centronics.

Atendendo a profissionais liberais e empresas de médio porte, a Microdigital desenvolveu o TK 2000 Color, compatível com a linha Apple, com RAM de 64 Kbytes e conexão paralela para impressora, além de saída para unidade de disquete. A versão melhorada (TK 2000 II) foi apresentada ao público no ano seguinte, 1985. A Microdigital oferece também alguns periféricos para esse modelo: kit de conexão para impressora, joystick, interface para unidade de disquete de 5 1/4 polegadas e interface serial RS232C, que permite a comunicação via rede telefônica com bancos de dados, como , Cirandão e o Videotexto.

Em junho de 1985, a Microdigital introduziu no mercado o TK 90X, compatível com o ZX Spectrum, da Sinclair. Nesse modelo, que incorpora interface para joystick e caracteres de acentuação para a língua portuguesa, destacam-se os inúmeros recursos que o tornam apropriado para aplicações educacionais, inclusive permitindo a utilização da linguagem LOGO.

Outro lançamento de sucesso foi o modelo de videogame Onyx Jr., compatível com a tecnologia Atari e de design simples e funcional.

Os produtos da Microdigital devem parte de seu sucesso à grande quantidade de softwares disponíveis, muitos dos quais desenvolvidos pela Microsoft. Há programas aplicativos, utilitários, jogos inteligentes, jogos animados, programas de engenharia e educacionais. Com o lançamento do TK 90X (um micro apropriado para implementação em escolas), a Microdigital evidencia sua intenção de abranger também o público estudantil. A empresa pretende apresentar os equipamentos ideais para quem se inicia na informática e implantar esse recurso de aprendizagem nos estabelecimentos de ensino.



Caminho aberto
Começando no canto inferior direito da tela, procure chegar à saída (canto superior esquerdo). À medida que você digita, indicando a direção, abre-se na tela um retângulo, que é o seu caminho.

```
205 REM
210 REM ** PRINT T$ AT X,Y **
215 FOR I=0 TO 3
220 YP=4*Y+I:P$=MID$(T$,4*I+1,4)
225 FOR L=0 TO 3
230 XP=4*X+L:C$=MID$(P$,L+1,1)
235 IF C$="1" THEN PLOT XP,YP
240 NEXT L,I
245 RETURN
250 REM ** INICIALIZA  **
255 HOME :GR :COLOR=12
260 FOR YP=4 TO 35
265 HLIN 4,35 AT YP
270 NEXT
275 COLOR=4
280 HLIN 8,36 AT 3:HLIN 3,36 AT 36
285 VLIN 3,36 AT 3:VLIN 3,36 AT 36
290 PX=8:PY=8:A(8,8)=1
295 X=PX:Y=PY
300 T$=RA$(1):COLOR=VAL(RA$(2)):GOSUB 21
    0
305 T$=PL$(1):COLOR=VAL(PL$(2)):GOSUB 21
    0
310 RETURN
315 REM ** MOVIMENTA **
320 REM
325 VTAB 1:GET IN$
330 DX=(IN$="K")-(IN$="J")
335 DY=(IN$="M")-(IN$="I")
340 IF DX=0 AND DY=0 THEN 325
345 ON A(PX+DX,PY+DY)+1GOTO 355,380
350 GOTO 325
355 REM
360 X=PX+DX:Y=PY+DY
365 T$=RA$(1):COLOR=VAL(RA$(2)):GOSUB 21
    0
370 A(X,Y)=1
375 RETURN
380 REM
385 X=PX:Y=PY
390 T$=RA$(1):COLOR=VAL(RA$(2)):GOSUB 21
    0
395 PX=PX+DX:PY=PY+DY
400 X=PX:Y=PY
405 T$=PL$(1):COLOR=VAL(PL$(2)):GOSUB 21
    0
410 IF PY>0 THEN RETURN
415 REM ** CHEGA AO FINAL **
420 HOME :VTAB 22:HTAB 1
425 PRINT "VOCE CONSEGUIU !!!"BE$;:END
430 REM ** MOVIMENTO DO MONSTRO **
435 MX=8:MY=8:X=MX:Y=MY
440 T$=MO$(1):COLOR=VAL(MO$(2)):GOSUB 21
    0
445 ON INT(RND(1)*2+1)GOSUB 520,525
450 LX=DX:LY=DY
455 REM ** ESCOLHE NOVA DIRECAO **
460 IF A(MX+LX,MY+LY)<1 THEN 480
465 IF A(MX+LY,MY-LX)=1 OR A(MX-LY,MY+LX
    )=1 THEN 500
470 REM ** MESMA DIRECAO **
475 DX=LX:DY=LY:GOTO 540
480 REM ** PANTANO A FRENTE **
485 ON INT(RND(1)*4+1)GOSUB 520,525,530,
    535
490 IF A(MX+DX,MY+DY)=-1 THEN 485
495 GOTO 540
500 REM ** POSSIBILIDADES **
505 ON INT(RND(1)*4+1)GOSUB 520,525,530,
    535
510 IF DX=-LX AND DY=-LY OR A(MX+DX,MY+D
    Y)<1 THEN 470
515 GOTO 540
520 DX=0:DY=-1:RETURN
525 DX=-1:DY=0:RETURN
530 DX=0:DY=1:RETURN
535 DX=1:DY=0:RETURN
540 REM ** EXECUTA MOVIMENTO **
545 X=MX:Y=MY
550 T$=RA$(1):COLOR=VAL(RA$(2)):GOSUB 21
    0
555 MX=MX+DX
560 MY=MY+DY
565 LX=DX:LY=DY
570 X=MX:Y=MY
575 T$=MO$(1):COLOR=VAL(MO$(2)):GOSUB 21
    0
580 IF MX=PX AND MY=PY THEN 620
585 IF A(MX,MY)=0 THEN 595
590 GOTO 455
595 REM ** MONSTRO AFOGA-SE **
600 A(MX,MY)=-1
605 X=MX:Y=MY
610 T$=DR$(1):COLOR=VAL(DR$(2)):GOSUB 21
    0
615 RETURN
620 REM ** JOGADOR INTERCEPTADO **
625 HOME :VTAB 22:HTAB 1
630 PRINT "DESCULPE-ME, MAS VOCE FOI COM
    IDO !"BE$;:END
635 REM ** PROGRAMA PRINCIPAL **
640 GOSUB 250
645 GOSUB 315
650 M=M+1
655 IF INT(RND(1)*(PX+PY))>3 OR M<5 THEN
    645
660 GOSUB 430
665 GOTO 645
670 REM ** INSTRUCOES **
675 TEXT :HOME :PRINT TAB(18)"ZUMBI"
680 VTAB 4:PRINT "VOCE ENCONTRA-SE EM UM
     PANTANO AS ESCU-"
685 PRINT :PRINT "RAS, E TEM QUE ACHAR S
    EU CAMINHO PARA"
690 PRINT :PRINT "FORA. MAS COMO DESGRAC
    A POUCA E' BOBA-"
695 PRINT :PRINT "GEM, ZUMBIS DO ALEM VA
    O A SUA PROCURA "
700 PRINT :PRINT "TENTANDO MATA-LO. BOA
    SORTE !!!"
705 PRINT :PRINT :INVERSE :PRINT "I";:NORMAL
    :PRINT "-CIMA       ";:INVERSE :PRINT
    "M";:NORMAL :PRINT "-BAIXO"
710 PRINT :INVERSE :PRINT "J";:NORMAL :PRINT
    "-ESQUERDA  ";:INVERSE :PRINT "K";:NORMAL
    :PRINT "-DIREITA"
715 VTAB 23:PRINT "QUALQUER TECLA PARA C
    ONTINUAR";
720 GET A$
725 RETURN
```

RASTREADOR
PARA LINHA APPLE

Sua missão é localizar e destruir um alvo o mais rápido possível. Para alcançar tal objetivo, você dispõe de um aparelho rastreador, que lhe indicará, por meio de sons, sua distância e razão de movimento. Seu veículo, porém, consome razoável quantidade de combustível e tem pequena autonomia. Assim, procure não perder tempo e evite desperdício de munição.
A tela informa a quantidade de combustível que resta nos tanques. Teclas que movimentam a nave: [I], para a frente; [M], para trás; [J], para a esquerda; e [K], para a direita. Quando quiser atirar, use a barra de espaço.

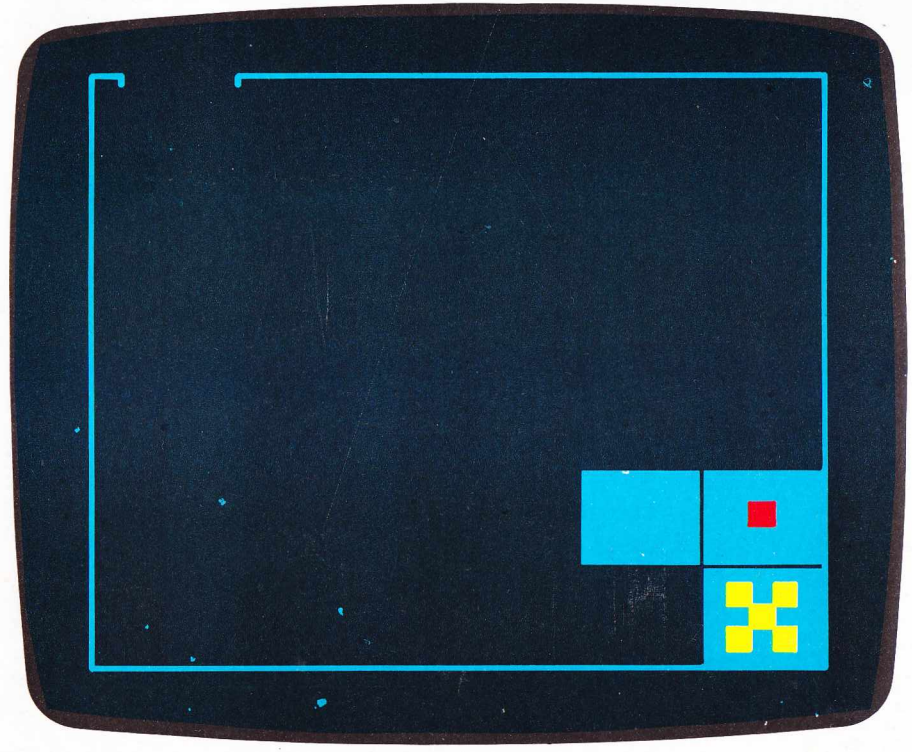
```
100 REM*********************************
105 REM*        PARA LINHA APPLE       *
110 REM*                               *
115 REM*                               *
120 REM*           RASTREADOR          *
125 REM*********************************
130 DIM X$(3),Y$(3),CO$(3)
135 NG=0:TR=0:MR=0
140 FOR I=1 TO 3
145 READ X$(I),Y$(I)
150 NEXT
155 DATA   E,C,"","",D,B
160 REM ** ROTINA DE SOM **
165 FOR X=770 TO 806
170 READ A
175 POKE X,A
180 NEXT
185 DATA  169,4,72,160,255,174,0,3,169,1
    ,32,168,252,202,208,6
190 DATA  173,48,192,174,0,3,136,208,239
    ,104,168,136,240,6,152
195 DATA  72,160,255,208,228,96
200 CO$(1)="DESCULPE-ME, MAS FOI MUITO L
    ONGE"
205 CO$(2)="ACHOU !!! PARABENS."
210 CO$(3)="ACABOU O COMBUSTIVEL."
215 GOTO 670
220 REM   ** INICIALIZACAO **
225 NG=NG+1
230 SX=INT(RND(1)*71-35)
235 SY=INT(RND(1)*71-35)
240 DI=ABS(SX)+ABS(SY)
245 MD=DI:SD=DI
250 EN=200:SH=0:HI=0:NS=0
255 VX=0:VY=0
260 RETURN
265 REM ** ENTRADA DE DADOS **
270 IN=PEEK(-16384)-128:POKE -16368,0
275 VX=3*((IN=ASC("K"))-(IN=ASC("J")))
280 VY=3*((IN=ASC("M"))-(IN=ASC("I")))
285 SH=(IN=ASC(" ")):REM ESPACO
290 RETURN
295 REM ** CALCULA DISTANCIAS, ETC. **
300 SX=SX-VX
305 SY=SY-VY
310 EN=EN-(VX<>0)-(VY<>0)-1
315 IF EN<=0 THEN EN=0:GOTO 375
320 REM ** ALVO SE AFASTA **
325 IF RND(1)<.5 THEN SX=SX+SGN(SX):GOTO
    335
330 SY=SY+SGN(SY)
335 DI=ABS(SX)+ABS(SY)
340 IF DI<MD THEN MD=DI
345 IF DI>100 THEN 375
350 IF NOT SH THEN 375
355 NS=NS+1
```

Bons ouvidos e reflexos
Rastreador é um jogo de ação que exige reflexos rápidos e boa audição. Para atingir o alvo você conta com um indicador de distância sonoro.

```
360 EN=EN-DI
365 IF EN<=0 THEN EN=0:GOTO 375
370 HI=(RND(1)*DI<5)
375 RETURN
380 REM ** SAIDA DE DADOS **
385 GOSUB 630
390 INVERSE
395 VTAB 7:HTAB 20+SGN(VX):PRINT X$(SGN(
    VX)+2);
400 VTAB 7+SGN(VY):HTAB 20:PRINT Y$(SGN(
    VY)+2);
405 NORMAL
410 IF NOT SH THEN 440
415 FOR I=1 TO 8
420 POKE 768,255
425 CALL 770
430 NEXT
435 GOTO 445
440 POKE 768,2*DI:CALL 770:REM  SOM
445 VTAB 1:HTAB 26:PRINT EN" ";
450 RETURN
455 REM  **  TELA  **
460 HOME
465 VTAB 1:HTAB 8:PRINT "COMBUSTIVEL:
      "EN" L."
470 GOSUB 630
475 VTAB 10:HTAB 17:PRINT "DIRECAO";
480 VTAB 17:HTAB 8:PRINT "DIGITE:
      I -CIMA"
485 HTAB 26:PRINT "J -ESQUERDA"
490 HTAB 26:PRINT "K -DIREITA"
495 HTAB 26:PRINT "M -BAIXO"
500 VTAB 22:HTAB 8:PRINT "ATIRE PRESSION
    ANDO ESPACO"
505 VTAB 24:HTAB 8:INVERSE :PRINT "QUALQ
    UER TECLA";:NORMAL
510 GET A$:HTAB 1:CALL -958
515 RETURN
520 REM ** FIM DESTE JOGO **
525 VTAB 3:HTAB 1:CALL -958
530 HTAB 8:PRINT CO$(CA)
535 FOR I=CA*10 TO CA*10+20 STEP 10
540 FOR K=I TO 50
545 POKE 768,K:CALL 770
550 NEXT K,I
555 PRINT :HTAB 8:PRINT "DISTANCIA INICI
    AL: "SD" M"
560 PRINT :HTAB 8:PRINT "NUMERO DE TIROS
    : "NS
565 PRINT :HTAB 8:PRINT "DISTANCIA MAIS
    PROX.  "MD" M"
570 IF HI THEN PRINT :HTAB 8:PRINT "VOCE
     O ACERTOU A "DI" M"
575 RE=50-DI/2:IF HI THEN RE=50+EN/4+SD/
    10
580 RE=INT(RE)
585 IF RE>100 THEN RE=100
590 IF RE<0 THEN RE=0
595 TR=TR+RE
600 IF RE>MR THEN MR=RE
605 PRINT :HTAB 8:PRINT "PONTUACAO EM UM
    A ESCALA DE 100:"
610 PRINT :HTAB 10:PRINT "ESTE JOGO:"TAB(
    32);RE
615 HTAB 10:PRINT "MEDIA EM "NG" JOGOS:
    "TAB(32);INT(TR/NG)
620 HTAB 10:PRINT "MAXIMO:"TAB(32);MR
625 RETURN
630 REM ** LIMPA DIRECAO **
635 INVERSE
640 VTAB 6:HTAB 20:PRINT " "
645 HTAB 19:PRINT " ";:HTAB 21:PRINT " "
650 HTAB 20:PRINT " ";
655 NORMAL
660 RETURN
665 REM ** PROGRAMA PRINCIPAL **
670 GOSUB 225
675 GOSUB 460
680 GOSUB 270
685 GOSUB 300
690 GOSUB 385
695 CA=3*(EN=0)+2*HI+(DI>100)
700 FOR W=1 TO 100:NEXT
705 IF CA=0 THEN 680
710 GOSUB 525:REM ** FIM **
715 POKE -16368,0
720 VTAB 23:HTAB 8:INPUT "OUTRO JOGO? ";
    A$
725 IF LEFT$(A$,1)="S" THEN 670
730 END
```


ZUMBI
PARA LINHA APPLE

Você está perdido num pântano em plena noite, à procura da saída. Para se locomover, precisa abrir caminho e andar para qualquer direção.
Digite [I] para ir em frente, [M] para voltar, [J] para virar à esquerda e [K] para a direita. Mas cuidado: à noite, há zumbis esfomeados rondando o pântano, seguindo seu rastro em busca de comida. Por sorte eles não têm visão muito boa e acabam se atolando no pântano. Por isso, nem sempre é aconselhável ir pelo caminho mais curto.



```
100 REM*********************************
105 REM*          PARA LINHA APPLE       *
110 REM*                                 *
115 REM*                ZUMBI            *
120 REM*********************************
125 DIM A(9,9),PL$(2),MO$(2),SW$(2),RA$(
    2),DR$(2)
130 BE$=CHR$(7)+CHR$(7):BE$=BE$+BE$+BE$
135 FOR C=0 TO 9
140 A(0,C)=-1:A(C,0)=-1
145 A(9,C)=-1:A(C,9)=-1
150 NEXT
155 A(1,0)=1
160 READ PL$(1),PL$(2)
165 READ MO$(1),MO$(2)
170 READ SW$(1),SW$(2)
175 READ RA$(1),RA$(2)
180 GOSUB 670:GOTO 635
185 DATA 0000011001100000,13
190 DATA 1001011001101001,1
195 DATA 1111111111111111,12
200 DATA 1111111111111111,7
```